On the Exact Solution of the No-Wait Flow Shop Problem with Due Date Constraints

Hamed Samarghandi
Department of Finance and Management Science, Edwards School of Business,
University of Saskatchewan, Saskatoon, Saskatchewan, Canada, S7N 5A7
samarghandi@edwards.usask.ca

Mehdi Behroozi
Department of Mechanical and Industrial Engineering, Northeastern University
334 Snell Engineering Center, 360 Huntington Avenue, Boston, MA 02115, United States
m.behroozi@neu.edu

Abstract

This paper deals with the no-wait flow shop scheduling problem with due date constraints. In the no-wait flow shop problem, waiting time is not allowed between successive operations of jobs. Moreover, the jobs should be completed before their respective due dates; due date constraints are dealt with as hard constraints. The considered performance criterion is makespan. The problem is strongly NP-hard. This paper develops a number of distinct mathematical models for the problem based on different decision variables. Namely, a mixed integer programming model, two quadratic mixed integer programming models, and two constraint programming models are developed. Moreover, a novel graph representation is developed for the problem. This new modeling technique facilitates the investigation of some of the important characteristics of the problem; this results in a number of propositions to rule out a large number of infeasible solutions from the set of all possible permutations. Afterward, the new graph representation and the resulting propositions are incorporated into a new exact algorithm to solve the problem to optimality. To investigate the performance of the mathematical models and to compare them with the developed exact algorithm, a number of test problems are solved and the results are reported. Computational results demonstrate that the developed algorithm is significantly faster than the mathematical models.

Keywords: No-Wait Flow Shop; Due Date Constraints; Mixed Integer Programming; Constraint Programming; Enumeration Algorithm

1 Introduction

In the classical flow shop scheduling problem there is a set of *n* jobs that has to be processed with a predefined order of operations on *m* machines, and the optimal sequence of jobs on each machine with respect to some performance measure is desired. It is also common to assume that jobs have identical sequence on all machines, which is known as the permutation flow shop scheduling problem.

Considered in this paper is a flow shop scheduling problem with the makespan criterion with two additional assumptions, namely allowing no waiting time between the operations and considering due date for each job. In the no-wait flow shop scheduling problem, no waiting time is allowed between successive operations of jobs. In other words, once processing of a certain job is started, no interruption is permitted between the operations of that job. In addition to the no-wait constraint we assume that the completion of each job is associated with a due date, i.e. jobs must be completed before their due dates. Due date are among the most applicable constraints in scheduling and sequencing literature because real-world jobs are usually accompanied by a deadline for completion (Hunsucker and Shah 1992). In this paper, it is assumed that all the jobs are ready at time zero (all release dates are zero) and no preemption or interruption in the process of operations is allowed. According to the conventional three-field notation of the scheduling problems (Graham et al. 1979), the problem can be designated as $F \mid nwt, d_j \mid C_{\max}$.

It has been shown by Wismer (1972) and Bonney and Gundry (1976) that the no-wait flow shop problem with makespan performance measure ($F \mid nwt \mid C_{\max}$) can be reduced to the asymmetric travelling salesperson problem (ATSP). Based on this relation between no-wait flow shop and ATSP, King and Spachis (1980) developed a heuristic to solve no-wait flow shop problems. It is proved by Lenstra and Kan (1979), using a reduction from the directed Hamiltonian path problem, that the problem $F \mid nwt \mid C_{\max}$ with *m* machines is NP-hard, when $m \geq 4$. They also showed that with $m = 2$ the problem is solvable in polynomial time. The NP-hardness of the case with $m = 3$ is shown by Röck (1984) using a reduction from three-dimensional matching (3DM) problem. Röck (1984) summarizes the complexity of a group of similar problems. Since $F \mid nwt \mid C_{\max}$ is a special case of $F \mid nwt, d_j \mid C_{\max}$ it can be concluded that $F \mid nwt, d_j \mid C_{\max}$ with at least three machines is also NP-hard in the strong sense.

Industrial applications mentioned in the literature for $F \mid nwt, d_j \mid C_{\max}$ include chemical industries (Rajendran 1994), food industries (Hall and Sriskandarajah 1996), steel production (Wismer 1972), pharmaceutical industries (Raaymakers and Hoogeveen 2000), and production of concrete products

(Grabowski and Pempera 2000). Hall and Sriskandarajah (1996) provide a comprehensive review of the applications of the problem.

The reputation of a company as a reliable firm will be tremendously damaged if it frequently delivers jobs after their due dates are passed (even if the number of late days is relatively small). Moreover, trust between companies will be damaged if late jobs are not frequent, but a few jobs are delivered considerably past their due dates. Note that on-time delivery of the jobs can be only one of the goals of a company. Companies can be interested in optimizing other criteria such as makespan, while avoiding late days or tardy jobs. Hence, $F \mid nwt, d_j \mid C_{\max}$ is not only an applicable problem with many real-world applications, but it is proved to be NP-hard and theoretically interesting.

The rest of the paper is organized as follows. Section 3 describes the notations used. Section 4 formulates the mathematical programming models. Section 5 describes the novel graph representation and the enumeration algorithm. Computational experiments are reported in section 6. Section 6.3 gives concluding remarks and discusses future research directions.

2 Related Work

The literature is rich with studies that develop heuristic or metaheuristic methods in order to deal with no-wait flow shop scheduling problems with or without due dates constraints. For the case of $F \mid nwt, d_j \mid \gamma$, due date constraints have been traditionally considered as soft constraints. In other words, violating due date constraints has been permitted with the objective function of minimizing a measure of the tardiness (e.g., number of tardy jobs or number of late days); tardiness measures have frequently been combined with other performance measures such as makespan, total flow time, etc.

Since no-wait flow shop problem with due date constraints is strongly NP-hard, several algorithms have been devised to deal with the problem. These efforts are reviewed in two categories, namely the heuristic or metaheuristic methods and exact methods, since both approaches can be useful depending on the size of the problems.

2.1 Heuristic and Metaheuristic Methods

Table 1 summarizes some of the early efforts to solve the scheduling problems with a form of due date constraints using a non-exact method. More recently, Tang et al. (2011) developed a metaheuristic to deal with fuzzy due dates in a flow shop environment. Panwalkar and Koulamas (2012) considered a two-machine flow shop problem with the objective of minimizing the total tardy jobs and finding a common due date for the jobs, and developed a heuristic algorithm with computational complexity of $O(n^2)$ for a

special case of the problem. This algorithm was further improved to an improved $O(n \log n)$ algorithm by Ilić (2015).

Aldowaisan and Allahverdi (2012) considered the flow shop scheduling problem with the objective of minimizing the number of tardy jobs and proposed a number of metaheuristics to deal with the problem. Arabameri and Salmasi (2013) considered the no-wait flow shop problem with the objective of minimizing the weighted earliness and tardiness penalties; they developed a MILP as well as a number of metaheuristics for the problem. The developed mathematical model of Arabameri and Salmasi (2013) is very similar to the model of Samarghandi (2015); however, it lacks some of the constraints of the model of Samarghandi (2015).

Table 1 - Early efforts to solve the flow shop problems with a form of due date constraints using a non-exact method

Reference	Addressed Problem	Objective Function	Solution Method
Rajasekera et al. (1991)	Flow shop	Due date assignment for the jobs	Queueing theory
Hunsucker and Shah (1992)	Multiple processor flow shop	Minimizing mean tardiness and number of tardy jobs	Simulation of the various priority rules
Sarper (1995)	Two-machine flow shop	Minimization of the sum of absolute deviation of job completion times from a common due date	Various heuristic methods
Brah (1996)	Flow shop and job shop	Minimizing mean tardiness and maximum tardiness	Comparison of various priority rules
Gupta et al. (2000)	Two-machine flow shop problem with a common due date	Minimizing earliness and tardiness	Heuristics derived from a branch and bound approach
Gowrishankar et al. (2001)	Flow shop	Minimizing the variance of completion times of jobs, and minimizing the sum of squares of deviations of job completion times from a common due date	Branch and bound and heuristics
Kaminsky and Lee (2002)	Flow shop	Due date assignment for the jobs; minimizing the sum of assigned due dates	Heuristics
Błażewicz et al. (2005)	Two-machine flow shop with common due date	Minimizing the total weighted late work	Dynamic programming
Błażewicz et al. (2008)	Two-machine flow shop with common due date	Minimizing the total weighted late work	Various metaheuristics
Hasanzadeh et al. (2009)	Two-machine flow shop with common due date	Minimizing the total weighted late work	Various metaheuristics
Dhingra and Chandna (2010)	Flow shop with sequence dependent setup times and due dates	Total weighted squared tardiness	Hybrid Genetic Algorithm (HGA)

Pang (2013) developed a genetic algorithm to deal with a two-machine no-wait flow shop problem with the objective of minimizing the maximum lateness of the jobs. Tasgetiren et al. (2013) considered the no-idle permutation flow shop scheduling problem with the total tardiness criterion and proposed an artificial bee colony algorithm to deal with the problem. Liu et al. (2013) proposed numerous heuristics for the no-wait flow shop problem with the objective of minimizing the total tardiness, and compared the results of the heuristics with each other.

Tari and Olfat (2014) considered a flow shop problem with due date constraints and proposed a number of heuristics to minimize the total tardiness. Ebrahimi et al. (2014) considered a hybrid flow shop problem in which each job is accompanied with an uncertain due date. The considered objective function is a combination of makespan and total tardiness; they proposed a number of metaheuristics to deal with this problem.

Ding et al. (2015) considered a no-wait flow shop problem with the objective of minimizing the total tardiness; they proposed a heuristic that is designed to speed up the search by focusing on a subset of the jobs rather than all the jobs. Fernandez-Viagas and Framinan (2015) studied a permutation flow show problem with a common due date for the jobs; they proposed two metaheuristics to deal with the problem and compared their results with the competitive methods. Shen et al. (2015) developed a metaheuristic to deal with the no-idle permutation flow-shop scheduling problem with the objective of minimizing the total tardiness.

Perez-Gonzalez and Framinan (2015) studied a permutation flow shop problem in which the jobs have a common due date. They considered two scenarios when a primary schedule is set up and new jobs arrive; first, to freeze the schedule and do not change it until all of the jobs are completed. Alternatively, to modify the schedule and accommodate the new jobs as long as the common due date is not violated. They performed computational experiments to determine which strategy works better when the jobs have certain characteristics.

Aldowaisan and Allahverdi (2012) and later on, Aldowaisan and Allahverdi (2015) considered the no-wait flow shop problem with due date constraints with the objective function of minimizing the total tardiness. They investigated the performance of various dispatching rules and introduced a simulated annealing and a genetic algorithm to deal with the problem.

Gupta and Kumar (2015) developed a heuristic algorithm to minimize a combination of total tardiness and makespan. Samarghandi (2015) studied the $F|nwt,d_j|C_{\max}$ and developed a particle swarm optimization to minimize the makespan; a Lagrangian relaxation method was proposed to deal with the violated due date constraints.

As one can notice, a common theme between all of the cited methods is that they first relax the due date constraints and then solve the remaining scheduling problem with a variant of the lateness/tardiness measure in the objective function by means of a heuristic or a metaheuristic algorithm.

2.2 Exact Methods

Mathematical programming techniques have long been employed to solve sequencing and scheduling problems. Selen and Hott (1986) developed a mixed integer programming for the flow shop problem. Stafford (1988) developed a mixed integer linear programming (MILP) based on the all-integer model of Wagner (1959).

Pekny and Miller (1991) compared the performance of an exact algorithm with a number of heuristic and metaheuristic algorithms developed for $F \mid nwt \mid \gamma$. This algorithm was initially introduced by Miller and Pekny (1991) to solve large-scale asymmetric travelling salesman problems. Later on, Pekny and Miller (1992) proposed a branch-and-bound algorithm for the ATSP to improve their previous algorithms.

Tseng et al. (2004) performed an empirical study to evaluate the performance of the different mixed integer programming (MIP) models for permutation flow shop problems; results of this study were in line with the results of Pan (1997) for the case of regular job shop and flow shop problems. Pan (1997) reported the models of Manne (1960), Wagner (1959), and Wilson (1989) as the first, second, and third best MILP formulations respectively; models developed by Bowman (1959), Gupta (1971), Morton and Pentico (2010), Baker and Baker (1974), and Stafford (1988) come next. It should be noted that these models are not reported in any special order.

More recently, Pan and Chen (2005) developed a mixed binary integer programming (MBIP) model for reentrant job shop scheduling problem. Ziaee and Sadjadi (2007) developed seven MBIP formulations for the flow shop sequencing problem and considered different constraints such as due dates, ready times, etc., and studied makespan, weighted mean flow time, and weighted mean tardiness as their performance measures. Javadi et al. (2008) developed a linear programming model for the no-wait flow shop problem with fuzzy objective functions.

Keha et al. (2009) reviewed the computational performance of four different MIP formulations for the single machine scheduling problem and its variants; they considered the due date constraints in some of these models. Finally, they introduced different sets of inequalities to improve these formulations. Ramezanian et al. (2010) developed a mathematical programming model to minimize the earliness and tardiness costs in a flow shop context, where processing times can be zero.

Demir and İşleyen (2013) compared the mathematical models developed for the flexible job shop problem; this problem can be considered as a generalization of $F \mid nwt \mid \gamma$. They noticed that certain types

of modeling generate better results; specifically, precedence variables usually require less CPU time, when being solved.

2.3 Contributions of the Paper

The above literature review clarifies that:

1. Due date constraints have rarely been studied as hard constraints. This is mainly due to the fact that generating a feasible solution for the problem, or proving that a feasible solution does not exist, turns into a very challenging task, especially when due dates are not too loose or too tight. This is true regardless of the employed solution methodology (Perez-Gonzalez and Framinan 2015).
2. Majority of the available methods in the literature of the flow shop problem with due date constraints are non-exact approaches. Hence, the need for developing effective and efficient exact methods for this problem, when the problem size justifies the use of exact algorithms, is deeply felt.
3. To the best of the authors' knowledge, there is no work in the literature that proposes an exact algorithm for the flow shop problem with no-wait and due date constraints.

As such, the main contributions of the current paper can be summarized as follows:

1. The due date requirements have been considered and dealt with as hard constraints, i.e., violation of the due dates is not allowed.
2. This study develops several mathematical programming formulations for $F \mid nwt, d_j \mid C_{\max}$. More specifically, an MIP, two quadratic MIPs, and two constraint programming (CP) models are developed. Baker and Keller (2010) reported that for the case of single machine sequencing problems mathematical programming models can be employed to optimally solve instances with as many as 50 jobs. However, computational experiments in this paper reveal that the number of jobs in $F \mid nwt, d_j \mid C_{\max}$ instances, which are normally more complex than single machine instances, should be smaller so that the problem can be solved to optimality using mathematical models.
3. This paper considers a new graph representation for the no-wait flow shop problem and proves a number of theorems based on the characteristics of the $F \mid nwt, d_j \mid C_{\max}$.
4. An enumeration algorithm is proposed to solve $F \mid nwt, d_j \mid C_{\max}$ to optimality; this algorithm employs the results of the proven propositions to shrink the feasible region of the problem and

to accelerate the search speed. Computational results reveal that the proposed algorithm is significantly faster than the discussed mathematical models.

3 Problem Description

In the considered $F|nwt,d_j|C_{\max}$ it is assumed that: 1) all jobs follow the same predefined order of operations; 2) no preemption or interruption is allowed; 3) no job can be processed by more than one machine at the same time, and no machine can process more than one operation at the same time; 4) all jobs must visit all machines with strictly positive processing time on all of the machines; and 5) there should be no waiting time between consecutive operations of a job. The following notation is used throughout the rest of this paper:

m	Number of machines
n	Number of jobs
J_j	Job j
p_{ij}	Processing time of i th operation of J_j
c_{jk}	Contribution of J_k to the objective function when placed immediately after J_j
S_{ij}	Starting time of i th operation of J_j
F_j	Finish time of J_j
d_j	Due date of J_j

A solution of $F|nwt|C_{\max}$ can be described with a sequence $\pi=(\pi_1,\pi_2,...,\pi_n)$ of n jobs. It is worth to be reminded that $F|nwt|C_{\max}$ is a permutation scheduling, i.e. the sequence of the jobs on all machines is the same. Hence, the contribution of job k when placed immediately after job j (c_{jk}) is not dependent to the machines. Contribution of J_j to $C_{\max}$ when J_j is the first scheduled job in a sequence is calculated as follows:

$$c_{0j}=\sum_{i=1}^{m}p_{ij};\ j=1,2,...,n \tag{1}$$

The algorithm of (Samarghandi (2015)) can be employed with small modifications to calculate $c_{jk};\ j,k=1,2,...,n;k\neq j$. Note that $c_{jj}=0;\ j=1,2,...,n$.

Step 1: Define a counter for the operations of π_j and a counter for operations of $\pi_k=\pi_{j+1}$; call the former counter t and the latter w.

Step 2: Set $t=2; w=1$.

Step 3: If $p_{tj} \geq p_{wk}$, set $t \leftarrow t+1$ and $w \leftarrow w+1$. If $t=m+1$, proceed to step 8; otherwise go back to the beginning of step 3. If $p_{tj} < p_{wk}$, proceed to step 4.

Step 4: Set $z=\left\{\min h \mid \left(\sum_{l=t}^{h} p_{lj}\right)-p_{wk} \geq 0\right\}$ and proceed to step 5. If the value of z cannot be determined, go to step 7.

Step 5: Set $p_{zj} \leftarrow \left(\sum_{l=t}^{z} p_{lj}\right)-p_{wk}$. Proceed to the next step.

Step 6: Set $w \leftarrow w+1$ and $t \leftarrow z$. If $t=m+1$, go to step 8; otherwise, go back to step 3.

Step 7: Set $c_{jk} \leftarrow \left(\sum_{l=w}^{m} p_{lk}\right)-\left(\sum_{l=t}^{m} p_{lj}\right)$. Stop.

Step 8: Set $c_{jk}=p_{mk}$. Stop.

The contribution matrix C is an $(n+1)\times n$ matrix that lists the contribution of each job to the makespan if placed after a certain job in the sequence.

$$C=[c_{jk}; j=0,1,...,n; k=1,2,...,n]=\begin{bmatrix} c_{01} & \cdots & c_{0n} \\ \vdots & \ddots & \vdots \\ c_{n1} & \cdots & c_{nn} \end{bmatrix} \quad (2)$$

The first row of C can be computed using (1). To calculate the rest of this matrix, the above algorithm should be used. Moreover, $c_{jj}=0; j=1,2,...,n$.

4 The Developed Models

This section presents the developed mathematical models.

4.1 Model I

The first model is based on the developed model of Samarghandi (2015) and employs the decision variable defined by (3). This model works directly with the problem data and does not require the algorithm of section 3 to calculate the contribution matrix.

$$x_{\substack{jk \\ j,k=1,2,...,n}} = \begin{cases} 1 & \text{if } J_k \text{ is placed immediately after } J_j \text{ in the sequence} \\ 0 & \text{Otherwise} \end{cases} \quad (3)$$

The model, which is a mixed integer programming, is as follows:

$$\text{minimize } C_{\max} \quad (4)$$

$$C_{\max} \geq S_{mj} + p_{mj}; \quad j = 1,2,...,n \quad (5)$$

$$S_{ik} + M(1 - x_{jk}) \geq S_{ij} + p_{ij}; \quad i = 1,2,...,m; \quad j,k = 1,2,...,n \quad (6)$$

$$S_{(i+1)j} = S_{ij} + p_{ij}; \quad i = 1,2,...,m-1; \quad j = 1,2,...,n \quad (7)$$

$$S_{mj} + p_{mj} \leq d_j; \quad j = 1,2,...,n \quad (8)$$

$$\sum_{j=1}^{n} x_{jk} \leq 1; \quad k = 1,2,...,n \quad (9)$$

$$\sum_{k=1}^{n} x_{jk} \leq 1; \quad j = 1,2,...,n \quad (10)$$

$$x_{jk} + x_{kj} \leq 1; \quad j,k = 1,2,...,n \quad (11)$$

$$\sum_{j=1}^{n} \sum_{k=1}^{n} x_{jk} = n - 1 \quad (12)$$

$$x_{jj} = 0; \quad j = 1,2,...,n \quad (13)$$

$$S_{ij} \geq 0; \quad i = 1,2,...,m; \quad j = 1,2,...,n \quad (14)$$

$$x_{jk} \in \{0,1\}; \quad j,k = 1,2,...,n \quad (15)$$

In this model, the objective function is to minimize the makespan; M is a sufficiently large number. (5) defines that makespan equals the finish time of the last operation of the last job. (6) assures that the operations do not overlap; this constraint is binding if J_k is scheduled immediately after J_j in the sequence. (7) imposes the no-wait constraints. (8) represents the due date constraint; according to (8), the last operation of each job should finish before its associated due date. Constraints (9), (10), (11), and (12) guarantee that all the jobs will appear exactly once in the sequence.

4.2 Model II

The sequence π is modified to include two dummy jobs, π_0 and π_{n+1} with zero processing times. Contribution matrix C of equation (2) is modified to C' to confirm that π_0 and π_{n+1} will be located in the first and the last positions in the sequence accordingly. In this matrix, $c_{jj} = 0; j = 1,2,...,n$.

$$C'_{(n+2)\times(n+2)} = [c_{jk}; j,k = 0,1,...,n+1] = \begin{bmatrix} 0 & c_{01} & \cdots & c_{0n} & 0 \\ 0 & \vdots & \ddots & \vdots & 0 \\ 0 & c_{n1} & \cdots & c_{nn} & 0 \\ M & 0 & \cdots & 0 & 0 \end{bmatrix} \tag{16}$$

$x_{jk}; j,k = 0,1,...,n+1$ is the binary decision variable of the model; $x_{jk} = 1$ indicates that J_k is placed immediately after J_j. If $x_{0k} = 1$, then J_k is the first job in the sequence. Accordingly, the following model is formulated.

$$\text{minimize } \sum_{j=0}^{n+1}\sum_{k=0}^{n+1} c_{jk}x_{jk} \tag{17}$$

$$\sum_{j=0}^{n} x_{jk} = 1; \quad k = 1,2,...,n+1 \tag{18}$$

$$\sum_{k=1}^{n+1} x_{jk} = 1; \quad j = 0,1,...,n \tag{19}$$

$$x_{j0} = 0; \quad j = 0,1,...,n+1 \tag{20}$$

$$x_{(n+1)k} = 0; \quad k = 0,1,2,...,n+1 \tag{21}$$

$$u_0 = 1 \tag{22}$$

$$2 \le u_j \le n+2; \quad j = 1,2,...,n+1 \tag{23}$$

$$u_j - u_k + 1 \le (n+1)(1 - x_{jk}); \quad j,k = 1,2,...,n+1; j \ne k \tag{24}$$

$$x_{jj} = 0; \quad j = 0,1,2,...,n+1 \tag{25}$$

$$F_0 = 0 \tag{26}$$

$$F_k = \sum_{j=0}^{n+1}(c_{jk} + F_j)x_{jk}; \quad k = 1,2,...,n+1 \tag{27}$$

$$F_j \le d_j; \quad j = 0,1,2,...,n+1 \tag{28}$$

$$x_{jk} \in \{0,1\}; \quad j,k = 0,1,...,n+1 \tag{29}$$

where (20) and (21) force the model to place the dummy jobs in their intended locations in the sequence. Equations (22), (23) and (24) are similar to the Miller-Tucker-Zemlin (MTZ) equations (Desrochers and Laporte 1991) and are used to avoid sub-tours when scheduling jobs in the sequence. According to (25) no job can be placed after itself. The recursive quadratic equation (27) calculates the finish time of J_k based on its predecessors. Due date constraints are enforced by (28). The following equations can be used to extract the sequence from the decision variables once the model is solved:

$$\pi_1 = \sum_{k=1}^{n} k x_{0k}$$

$$\pi_j = \sum_{k=1}^{n} k x_{\pi_{(j-1)},k}; \quad j = 2,3,...,n$$

4.3 Model III

Although this model employs the same contribution matrix as Model I and Model II, the decision variable of this model, is defined as follows (as there are n jobs and n possible locations in the sequence):

$$\underset{l,j=1,2,...,n}{x_{lj}} = \begin{cases} 1 & \text{if } \pi_l = J_j \\ 0 & \text{otherwise} \end{cases} \tag{30}$$

In this model, L_l is a variable that will be used to calculate the finish time of π_l. Based on this definition for the decision variables, the model can be formulated as:

$$\text{minimize } L_n \tag{31}$$

$$\sum_{l=1}^{n} x_{lj} = 1; \quad j = 1,2,...,n \tag{32}$$

$$\sum_{j=1}^{n} x_{lj} = 1; \quad l = 1,2,...,n \tag{33}$$

$$\sum_{l=1}^{n} \sum_{j=1}^{n} x_{lj} = n \tag{34}$$

$$L_1 = \sum_{j=1}^{n} c_{0j} x_{1j} \tag{35}$$

$$L_l = \sum_{j=1}^{n} \sum_{\substack{k=1 \\ k \neq j}}^{n} x_{(l-1)j} x_{lk} c_{jk} + L_{l-1}; \quad l = 2,3,...,n \quad (36)$$

$$L_l \leq \sum_{j=1}^{n} d_j x_{lj}; \quad l = 1,2,...,n \quad (37)$$

$$L_l \geq 0; \quad l = 1,2,...,n \quad (38)$$

$$x_{lj} \in \{0,1\}; \quad l, j = 1,2,...,n \quad (39)$$

In this model, (31) minimizes the makespan by minimizing the finish time of π_n. (35) calculates the finish time of π_1; the first term of (36) calculates the contribution of J_k to the makespan when it is located after J_j. (37) is the due date constraint.

Model III is formulated based on the finish time of the jobs in different positions; finish times were calculated by equations that were independent from the job that is located in each position. However, it is possible to modify Model III to calculate the finish times of the jobs rather than the finish times of the positions. In Model III, L_l is calculated by searching the rows of the C' matrix. In the modified model, finish time calculations are performed by exploring both the rows and the columns of C'. Assume that F_j is the finish time of J_j. Therefore, in the modified model, equations (35) and (36) should be replaced with the following:

$$F_{k \atop k=1,2,...,n} = \begin{cases} x_{1k} c_{0k} & \text{if } x_{1k} c_{0k} > 0, k = 1,2,...,n \\ \sum_{l=2}^{n} \sum_{\substack{j=1 \\ j \neq k}}^{n} x_{(l-1)j} x_{lk} c_{jk} + \sum_{l=2}^{n} \sum_{\substack{j=1 \\ j \neq k}}^{n} x_{(l-1)j} x_{lk} F_j & \text{otherwise} \end{cases} \quad (40)$$

The first condition of (40) is true only for π_1. All the other jobs will utilize the second condition. Finish time of J_k dependents on the finish time of its immediate predecessor J_j. Once the finish times are defined by (40), the objective function of Model III and the due date constraints will be modified accordingly:

$$\text{minimize} \quad \max_j F_j$$

$$F_j \leq d_j; \quad j = 1,2,...,n$$

In the modified model equations (35) and (36) should be replaced with (40), which is a quadratic non-convex equation. This makes the model complicated and difficult to solve. Therefore, although the modified model is of theoretical interest, it will not be further investigated for the computational experiments.

4.4 Model IV

Unlike previous models, Model IV and Model V are formulated based on the special characteristics and properties of constraint programming (CP). The decision variable that will be used for Model IV and Model V is defined as $x_l = j$ if J_j is placed in location l; one should define $x_0 = 0$. The contribution of the jobs to the makespan is defined the same way as in the previous models, which is based on placing a certain job after another job; however, for Model IV and Model V it is assumed that $c_{jj} = M; j = 1,2,...,n$ (M is a sufficiently large number). This will prevent the CP model from placing a certain job after itself. Accordingly, the first CP model will be as follows:

$$\text{minimize} \quad c_{0,x_1} + \sum_{l=2}^{n} c_{x_{l-1},x_l} \tag{41}$$

$$\text{All Different}(x_1, x_2, ..., x_n) \tag{42}$$

$$\sum_{l=1}^{j} c_{x_{(l-1)},x_l} \le d_{x_j}; \quad j = 1,2,...,n \tag{43}$$

$$x_l \in \{1,2,...,n\}; \quad l = 1,2,...,n \tag{44}$$

The objective function is defined based on the contribution of the jobs once the sequence is determined. The combination of (42) and (44) guarantees that all the jobs will be placed in the sequence, and each job will appear in the sequence only once. (43) is the due date constraint; finish times of the jobs are calculated based on the contribution of the previous jobs in the sequence.

4.5 Model V

This model is based on the same decision variable as Model IV. However, Model V unlike Model IV, works directly with the problem data and therefore, does not require the contribution matrix.

$$\text{minimize} \quad S_{m,x_n} + p_{m,x_n} \tag{45}$$

$$\text{All Different}(x_1, x_2, ..., x_n) \tag{46}$$

$$S_{i,x_{(j+1)}} \ge S_{i,x_j} + p_{i,x_j}; \quad i = 1,2,...,m; \quad j = 1,2,...,n-1 \tag{47}$$

$$S_{(i+1),x_j} = S_{i,x_j} + p_{i,x_j}; \quad i = 1,2,...,m-1; \; j = 1,2,...,n \tag{48}$$

$$S_{m,x_j} + p_{m,x_j} \le d_{x_j}; \quad j = 1,2,...,n \tag{49}$$

$$S_{ij} \ge 0; \quad i = 1,2,...,m; \; j = 1,2,...,n \tag{50}$$

$$x_j \in \{1,2,...,n\}; \quad j = 1,2,...,n \tag{51}$$

In this model, (47) means that the jobs should not overlap. (48) represents the no-wait constraints and (49) belongs to the due date constraints. The enumeration algorithm will be presented in the next section. A comparison between Model IV and Model V is presented in Table 4.

4.6 Model Validation and Efficiency Comparison

The validation of the proposed models has been tested by: 1) a comparative analysis of the models with the existing models for $F \| C_{\max}$ and $F \mid nwt \mid C_{\max}$ problems and also with one another, and 2) thoroughly checking the assumptions, definitions, variables, and constraints in both subjective and objective ways. Moreover, the correctness of the models has been verified by their ability to find the optimal solutions of the standard benchmark problems from the OR-Library[1].

In order to comparatively analyze the computational efficiency of the models, a comparison between the number of variables and constraints of Model I, Model II, and Model III is presented in Table 2. Among these models, Model I can be compared to the classical models of the flow shop problem by removing the due date and no-wait constraints. This is impossible for Model II and Model III, since they follow a different logic and are based on the concept of contribution matrix. Pan (1997) reports the number of variables and constraints for some of the classical models and states that Manne model is the best among them. Table 3 compares the computational efficiency of Model I with best three of the models mentioned in Pan (1997) that includes Manne (1960), Wilson (1989), and Wagner (1959) as well as two other models, Stafford (1988) and Šeda (2007). In section 6, Manne model, as one of the best and most commonly used models, is used again to verify the performance of the developed models and algorithms on the test problems. This also serves as another tool for model validation and verification purposes. Table 5 compares the computational efficiency of the two constraint programming models, Model IV and Model V.

[1] Retrieved June 04, 2016, from http://people.brunel.ac.uk/~mastjjb/jeb/info.html

Table 2 - Comparison of the MIP models for $F \mid nwt, d_j \mid C_{\max}$

	Number of binary variables	Number of continuous variables	Total number of constraints	Number of due date constraints
Model I	n^2	mn	$(m+1)n^2+(m+4)n+1$	n
Model II	$(n+2)^2$	$n+2$	$n^2+8n+14$	$2n+4$
Model III	n^2	n	$4n+1$	n

Table 3 - Comparison of the MIP models for $F \| C_{\max}$

	Number of binary variables	Number of continuous variables	Total number of constraints
Model I – without due date and no-wait constraints	n^2	mn	$(m+1)n^2+4n+1$
Wagner (1959)	$n^2(m-2)$	$2mn+1$	$n^3(m-1)+3mn$
Manne (1960)	$\frac{(m-2)n(n-1)}{2}$	$\frac{mn(n+1)}{2}+1$	$\frac{mn(n+1)}{2}$
Stafford (1988)	n^2	$(2m-1)n$	$n+m(n-1)+1$
Wilson (1989)	$n^2(m-2)$	$mn+1$	$n^3(m-1)+3mn-m+1$
Šeda (2007)	n^2	$2mn$	$mn+m+n$

Table 4 - Comparison of the constraint programming models

	Number of integer variables	Number of continuous variables	All different constraints	Number of due date constraints	Number of other constraints
Model IV	n	0	1	n	0
Model V	n	mn	1	n	$2mn-m-n$

5 A Novel Graph Representation and Enumeration Algorithm

In this section a new graph representation of the problem $F \mid nwt, d_j \mid C_{\max}$ is proposed. In the rest of the paper this graph will be called search graph. The search graph enables us to exploit the characteristics of the solution set of the problem $F \mid nwt, d_j \mid C_{\max}$ and use them to efficiently develop the enumeration algorithm by removing a large chunk of uncompetitive solutions. Figure 1 describes a search graph that represents the $F \mid nwt, d_j \mid C_{\max}$:

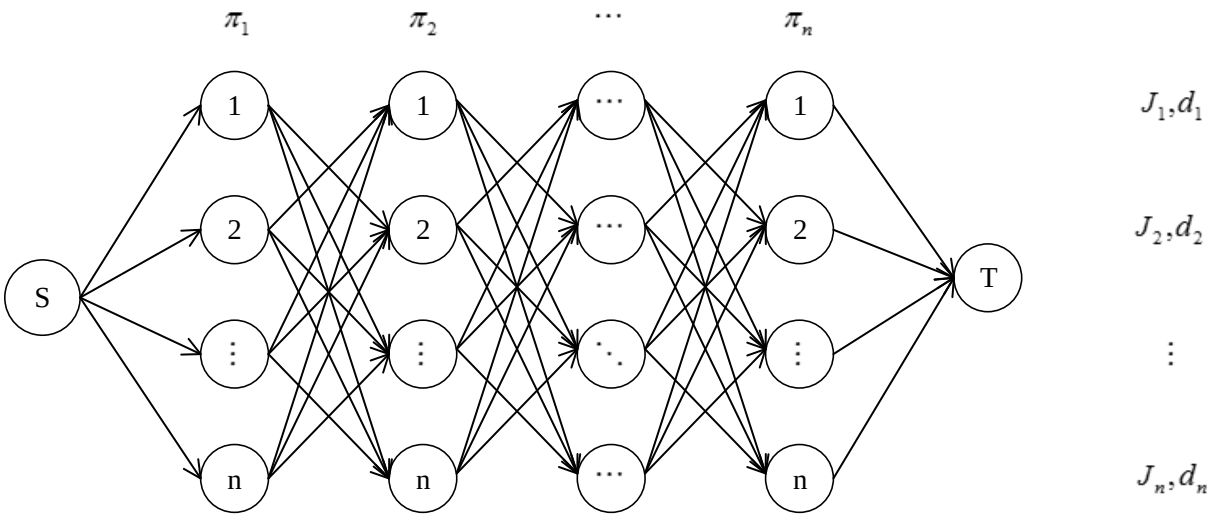

Figure 1 - The search graph respresenting $F \mid nwt, d_j \mid C_{\max}$

Let $G = \{V, E\}$ denote the search graph, which contains n rows and columns. It is easy to see that *G* has $|V| = n^2 + 2$ nodes; each node if located in the intersection of row $j; 1 \le j \le n$ and column $l; 1 \le l \le n$ represents job j if located in position π_l of permutation π. Nodes S and T are dummy jobs with zero processing times, which represent the start and the finish of the flow shop system. An arc exists between two nodes if and only if these nodes belong to two adjacent columns and they do not represent the same job; as a result, the number of arcs between two adjacent columns are $n(n-1)$ and the total number of arcs are $n(n-1)^2$. Arcs that start from node S or end at node T are excluded in the above calculations. Figure 2 describes an instance of $F \mid nwt, d_j \mid C_{\max}$ with three jobs.

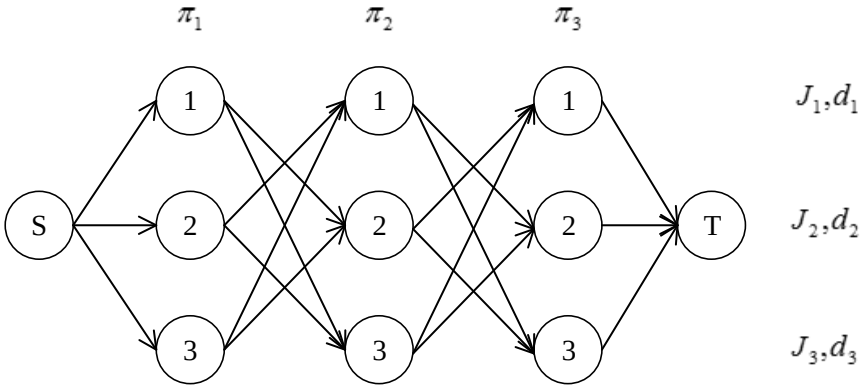

Figure 2 - An instance of $F \mid nwt, d_j \mid C_{\max}$

5.1 Feasible Solution of $F\,|\,nwt\,|\,C_{\max}$ In a Search Graph

A feasible solution of $F\,|\,nwt\,|\,C_{\max}$ starts with S and ends with T; it includes one and only one node in each row and in each column. As a result, Figure 3 characterizes the permutation $\pi=(2,1,3)$ and represents a feasible solution of $F\,|\,nwt\,|\,C_{\max}$ with three jobs.

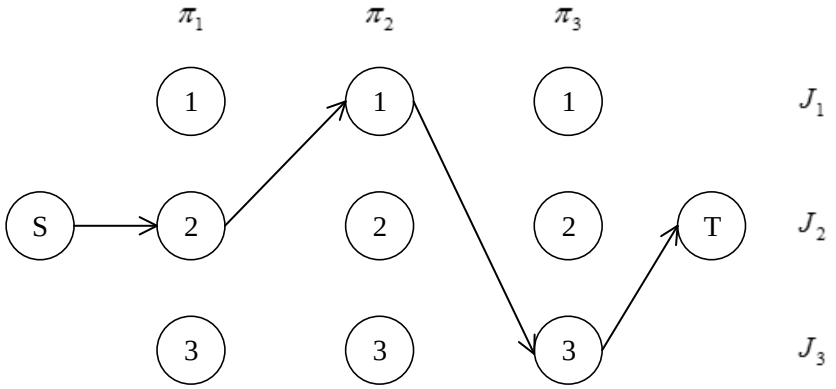

Figure 3 – A feasible solution of $F\,|\,nwt\,|\,C_{\max}$ with three jobs and three machines

Each arc $a_{jk};1\le j,k\le n$, when a_{jk} exists, can be labeled with c_{jk} as defined by (2). a_{Sj} represents the arc that connects S to J_j in column π_1 and is labeled with c_{0j} defined by (1). As a result, for Figure 3, the makespan is as follows:

$$C_{\max}=c_{02}+c_{21}+c_{13} \tag{52}$$

It can be noted that the permutation $\pi=(2,1,3)$ in Figure 3 is a feasible solution of $F\,|\,nwt,d_j\,|\,C_{\max}$ if:

$$\begin{aligned} &c_{02}\le d_2\\ &c_{02}+c_{21}\le d_1\\ &c_{02}+c_{21}+c_{13}\le d_3 \end{aligned} \tag{53}$$

Moreover, if $\pi=(2,1,3)$ is the shortest path from S to T, π is the optimum solution of the $F\,|\,nwt,d_j\,|\,C_{\max}$ instance which is described in Figure 3. It can be verified that the number of permutations for an instance of $F\,|\,nwt,d_j\,|\,C_{\max}$ with n jobs and m machines, as described by Figure 1, is $n!$.

Observation 1. Suppose that LP_{j,π_l} represents the longest path from S to the node in the intersection of column π_l and row j. If $LP_{j,\pi_l} \le d_j; \forall j \in \{1,2,...,n\}, \forall l \in \{1,2,...,n\}$, then the due date constraints can be removed and the problem reduces to $F \mid nwt \mid C_{\max}$.

Observation 2. If $LP_{j,\pi_n} \le d_j; \forall j \in \{1,2,...,n\}$, then the due date constraints can be removed and the problem reduces to $F \mid nwt \mid C_{\max}$.

Observation 3. If $\exists j \in \{1,2,...,n\} \mid LP_{j,\pi_n} \le d_j$, then the due date constraints for J_j can be removed from the problem.

Observation 4. Suppose that SP_{j,π_l} represents the shortest path from S to the node in the intersection of column π_l and row j. If $\exists j \in \{1,2,...,n\} \mid SP_{j,\pi_l} > d_j, \forall l \in \{1,2,...,n\}$, then the problem is infeasible. If $\exists j \in \{1,2,...,n\} \mid SP_{j,\pi_l} > d_j, \forall l \in \{1,2,...,n\}$ or $\exists l \in \{1,2,...,n\} \mid SP_{j,\pi_l} > d_j, \forall j \in \{1,2,...,n\}$, then the problem is infeasible.

5.2 Eliminating Infeasible Solutions

In order to shrink the size of the set of solutions to enumerate to find the optimal solution, the following results are useful.

Observation 5. Due to the no-wait constraints, any feasible solution of $F \mid nwt \mid C_{\max}$ with $p_{ij} > 0, \forall i, j$ is a permutation schedule, i.e. the order of jobs on all machines remains the same.

Observation 6. For $F \mid nwt \mid C_{\max}$, any non-semi-active feasible schedule can be easily transformed to a semi-active feasible schedule considering the no-wait constraint, with the same or a better objective function value. This can be done by simply removing the non-necessary delays for all operations without changing the sequence or violating the no-wait constraints.

Observation 7. For any two consecutive jobs in a semi-active feasible solution of $F \mid nwt \mid C_{\max}$, there exists at least one machine with no idle time between processing of the operations of these two jobs, otherwise the solution would not be semi-active.

Proposition 1. For $F \mid nwt \mid C_{\max}$ with $p_{ij} > 0, \forall i, j$ with a non-empty feasible set, the set of semi-active feasible schedules and the set of active feasible schedules are non-empty and equal.

Proof. By **Observation 6** it is clear that as long as the set of feasible solutions is not empty, then the set of all semi-active schedules is non-empty. Since the set of all active schedules is a subset of the set of all semi-active schedules, it is enough to prove that each semi-active schedule is also active. Due to the no-wait constraints and **Observation 5** and **Observation 7**, it is impossible to construct a new schedule, through reordering the sequence, with at least one operation finishing earlier without delaying another operation. Hence any semi-active schedule is also active.

Corollary 1. There exists for $F \mid nwt \mid C_{\max}$ an optimal schedule that is active considering the no-wait constraints.

Proposition 2. For an active feasible solution of $F \mid nwt \mid C_{\max}$ with the partial permutation $(..., j, k, ..., q, ...)$, it can be proved that $c_{jk} < c_{jq} + c_{qk}$.

Proof. The proof is by contradiction. Assume that this is not true; then $c_{jk} \geq c_{jq} + c_{qk}$. Let $C^{j}_{\max}$ be the objective function of the partial solution $\pi = (..., j)$; then $C^{j}_{\max} + c_{jk} \geq C^{j}_{\max} + c_{jq} + c_{qk}$. This means by scheduling job q between job j and job k the finish time of job k (F_k) must either remain the same or be reduced by some positive amount. In either case, none of the operations of job k will be delayed since there is no waiting time between the operations of a job. This means that one is able to schedule job q between job j and job k without delaying any of the operations of job k. This contradicts the assumption of the solution being active.

Corollary 2. Given a partial permutation π for $F \mid nwt \mid C_{\max}$ with $F_j \leq d_j, \forall j \in \pi$, if constructing the partial permutation $\pi' = (\pi, k)$ for some k results in $F_k > d_k$, then any permutation of the form $\pi'' = (..., \pi, ..., k, ...)$, which places k after π, is infeasible.

Proof. Finish time of each job is the sum of the contribution of the jobs in the partial sequence ending to that job. Therefore by **Proposition 2**, F_k will be increased by placing more jobs between π and job k. Among all permutations that place job k after π, the permutation $(\pi, k, ...)$ will have the smallest F_k which is still infeasible.

Observation 8. If $\exists l \in \{2, 3, ..., n\}, j \in \{1, 2, ..., n\} \mid SP_{j,\pi_l} > d_j$, then it is possible to remove this node as well as all of the arcs that start from or end at this node from G. In other words, by placing this job in

location π_l of the permutation, the due date constraints will be violated. Removing a node in column π_1 means that the problem is infeasible; removing a node in column $\pi_l; 2 \le l \le n-1$ results in the removal of $2(n-1)$ arcs from G; removing a node from column n results in the removal of n arcs from G.

5.3 The Enumeration Algorithm

Algorithm 1. The following algorithm represents the enumeration algorithm that solves $F \mid nwt, d_j \mid C_{\max}$ to optimality.

1. If $\exists j \in \{1,2,...,n\} \mid SP_{j,\pi_1} > d_j$, stop. The problem is infeasible.

2. If $LP_{j,\pi_n} \le d_j; \forall j \in \{1,2,...,n\}$, remove the due date constraints to reduce the problem to $F \mid nwt \mid C_{\max}$.

3. Calculate $SP_{j,\pi_l}; l \in \{2,3,...,n\}, j \in \{1,2,...,n\}$. If $\exists j \in \{1,2,...,n\} \mid SP_{j,\pi_l} > d_j; l \in \{2,3,...,n\}$, remove the corresponding node and all of its arcs from the graph G; call the remaining graph G'.

4. Find the shortest path between S and T with attention to the definition of the feasible solution of $F \mid nwt \mid C_{\max}$. If the found shortest path does not violate any of the due date constraints, it is optimal; compute the total contribution values of this path to calculate the makespan. Otherwise, proceed to step 5.

5. This step describes an enumeration sub-algorithm to solve G' to optimality. The objective of this sub-algorithm is to fathom all of the paths of the modified search graph (or G') from S to T until the optimum solution is found. The root node is S.

5.1. Branch from S to all of the nodes in π_1. Define l as the index for the positions in the permutation; in other words, l represents the current column in G'. Set $l \leftarrow 1$. Objective function value for node $j; j \in \{1,2,...,n\}$ is $C_j^l = c_{0j}$. Fathom all nodes in G' for $l > 1$.

5.2. Assume that $C_q^l = \max_j \left\{ C_j^l \mid j = 1,2,...,n;\ j \text{ is not selected or fathomed yet} \right\}$; update the current node to q; break the ties by random selection, unfathom all the nodes in column $t \mid t > l$, and branch from q to all of its adjacent nodes in G'; calculate $C_j^{l+1} \leftarrow C_q^l + c_{qj}; j \in \{1,2,...,n \mid q \text{ and } j \text{ are adjacent}\}$.

5.3. Fathom the nodes that violate the due date of their respective jobs in column $l+1$, and go to step 5.6 if $l \ne n-1$; otherwise proceed to step 5.4. Note that if due date constraints are violated when $l = 1$, according to step 1 the problem is infeasible.

5.4. Compare $C_j^{l+1}; j \in \{1,2,...,n\}$ with $C_{\max}^{best}$, the makespan of the best-known feasible solution (if the list of the complete feasible solutions is not empty); if $C_j^{l+1} > C_{\max}^{best}; j \in \{1,2,...,n\}$, fathom node j in column $l+1$.

5.5. If $l = n-1$ and there is at least one node in column $l+1$ which is not fathomed yet, then the paths to such nodes define different feasible solutions each with makespan which is at least as desirable as $C_{\max}^{best}$. Accordingly, compare the makespan of such nodes with each other and update $C_{\max}^{best}$ with the best found makespan. Then, fathom all the nodes in column $l+1$ and proceed to 5.6.

5.6. If all of the nodes in $l+1$ are fathomed, then fathom the current node and proceed to 5.6.1. Otherwise, set $l \leftarrow l+1$ and go to step 5.2.

5.6.1. If there are nodes in the current column l, which have not yet been selected or fathomed during the course of the algorithm, do not change the value of l; go to step 5.2. Otherwise proceed to 5.6.2.

5.6.2. Set $l \leftarrow l-1$. If $l = 0$, stop. Report $C_{\max}^{best}$ and its corresponding route as the optimum solution. If the list of the feasible solutions is empty, the problem is infeasible. Otherwise, restart step 5.6 from the beginning. ■

Figure 4 illustrates the enumeration sub-algorithm. Note that the above algorithm does not exploit the results of **Corollary 2**. In order to integrate **Corollary 2** in the algorithm, steps 5.3 and 5.4 of **Algorithm 1** should be modified as follows; this results in **Algorithm 2**. The rest of the steps remain unchanged.

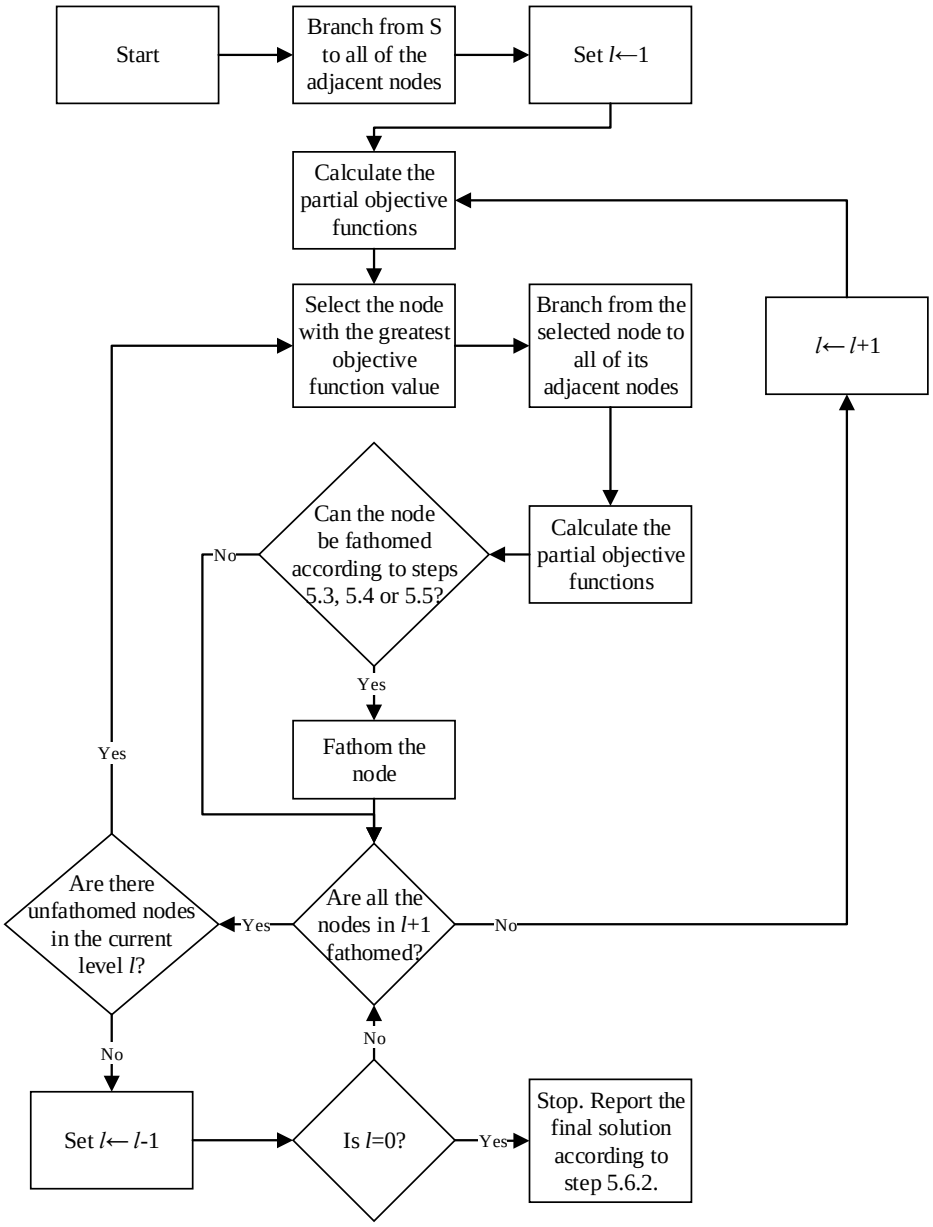

Figure 4 – Flow chart of the Algorithm 1

Algorithm 2. Modify steps 5.3 and 5.4 of **Algorithm 1** as follows:

5.3′ Fathom all the nodes in column $l+1$; if $l \neq n-1$, then go to step 5.6. Otherwise, proceed to step 5.4′. Note that if due date constraints are violated when $l=1$, according to step 1 of **Algorithm 1** the problem is infeasible.

5.4.′ Compare $C_j^{l+1}; j \in \{1,2,...,n\}$ with $C_{\max}^{best}$, the makespan of the best-known feasible solution (if the list of the complete feasible solutions is not empty); if $C_j^{l+1} > C_{\max}^{best}; j \in \{1,2,...,n\}$, fathom all the nodes in column $l+1$. ■

5.4 Proof of the Correctness of the Enumeration Algorithms

Algorithm 1 and Algorithm 2 are correct because:

1. The $F \mid nwt, d_j \mid C_{\max}$ with the assumptions of section 3 is a permutation problem, i.e., a sequence of the jobs is enough to describe a potential solution. Makespan of a given permutation can be calculated by adding the contributions of the jobs. Section 3 also proposed as a method for calculating the contributions. This validates the graph modeling of section 5.1.

2. Step 5 of Algorithm 1 is designed to generate all the possible $n!$ permutations of n jobs in a $F \mid nwt, d_j \mid C_{\max}$.

2.1. Steps 5.1, 5.2 and 5.6 describe the branching strategy of the enumeration algorithm; they explain how a sequence of n jobs will be created. In other words, steps 5.1 and 5.2 create all of the possible solutions, and step 5.6 explains how the algorithm moves from one sequence to another permutation.

3. The proposed algorithm does not ignore any feasible solutions throughout the search. Only the infeasible or uncompetitive solutions will be removed:

3.1. Step 5.3 mentions that a sequence of jobs that violates the due date requirements of some of the jobs is considered exempt from further consideration.

3.2. Step 5.4 assures that a partial sequence with a makespan worse than the makespan of a complete solution is exempt from further consideration. This is because the makespan can never be reduced by adding more jobs to a partial sequence. In other words, contributions are always non-negative.

4. The proposed algorithm compares the makespan of all of the feasible solutions, and returns the best makespan as the final solution after all of the feasible solutions have been considered:

4.1. Step 5.5 compares the makespan of a complete sequence with the makespan of the best-found sequence, and updates the best-found sequence and its makespan if necessary.

5. Rest of the steps in Algorithm 1 limit the enumeration as much as possible and are based on the proved propositions of section 5.

6. The described enumerative procedure guarantees that all of the feasible solutions are considered and the optimum solution is found before the termination of the algorithm.

7. Algorithm 2 simply strengthens steps 5.3 and 5.4 of the Algorithm 1 based on Proposition 1 and Corollary 2 of section 5.

Numerical results will be presented in the next section.

6 Computational Experiments

Conducting numerical experiments is an effective approach to compare the performance of the developed models. IBM ILOG CPLEX V12.6 was used to solve the developed mathematical models. Algorithms of Section 5 were coded by Microsoft Visual C++ 2013. All the numerical experiments were performed on a PC equipped with a 2GHz Intel Pentium IV CPU and 2 GB of RAM. To perform the computational analysis, a number of test problems generated by Samarghandi (2015) were selected; namely, eight test problems for $F\,|\,nwt\,|\,C_{\max}$ accompanied with four different due date settings for each test problem. Moreover, 26 other test problems with larger instances for $F\,|\,nwt\,|\,C_{\max}$ were generated. Each test problem was then accompanied by four different due date settings. All the test problems were generated based on the same approach described by Samarghandi (2015). Accordingly, a total of 104 test problems for $F\,|\,nwt, d_j\,|\,C_{\max}$ and 14 test problems for $F\,|\,nwt\,|\,C_{\max}$ were investigated in this paper; each distinct due date setting will be called a tightness factor and will be abbreviated as TF hereinafter. *Sam01* through *Sam08* are test problems for $F\,|\,nwt\,|\,C_{\max}$ from Samarghandi (2015) and *Sam01+DD* through *Sam08+DD* are test problems with due date constraints from Samarghandi (2015); problems generated in this study are *Sam09* through *Sam14* and *Sam09+DD* through *Sam26+DD*.

Best solutions of the models for the test problems will be reported at $T = 60$, $T = 300$, $T = 600$ and $T = 7200$ seconds. Before the results are presented, some of the complications when solving the problems will be discussed.

6.1 Implementation Complications

Formulation of Model I is based on a very large number (M) in (6) that replicates either-or constraints. Although this is an effective method to prototype either-or constraints, the numerical value of M may result in complication in implementation of the model in any software package designed for solving mathematical modeling problems; IBM CPLEX is not an exception. If the value of M is not carefully chosen, CPLEX may eliminate M in the pre-solve phase. It is therefore recommended[2] that either-or constraints should be modeled by indicator constraints in order to eradicate the need for the numerical value of M . However, employing indicator constraints results in a reduction in the effectiveness of the branching

[2] http://www-01.ibm.com/support/docview.wss?uid=swg21400084

algorithm; this can result in an increase in the solution time. Numerical results of both of these approaches to implement Model I will be presented in section 6.2.

6.2 Numerical Results of the Developed Models

The equality (27) in Model II is a quadratic equation, which makes it a non-convex constraint. The same argument holds for equation (36) in Model III. Hence solving these two models even after relaxing the integrality constraint is not easy. There is a bulk of research on finding approximate solutions for non-convex binary integer programming using convex optimization techniques like SDP relaxation (see e.g. the pioneering paper of Goemans and Williamson (1995) on MAX-CUT Problem). However, this paper does not seek approximate solutions so the authors have taken this problem as an interesting future research direction. For this reason, in this paper Model II and Model III will not be included in the numerical experiments for $F\,|\,nwt, d_j\,|\,C_{\max}$.

On the other hand, in order to review the performance of Model II, the due date constraints of this model, including the non-convex constraint, will be relaxed and computational experiments will be conducted for $F\,|\,nwt\,|\,C_{\max}$ and compared with the no-wait version of the Manne model, which simply is the Manne model with the no-wait constraints added to it, as well as the relaxed version of Model I, Model IV and Model V. The reason for selecting the Manne model as a comparison basis is that Pan (1997) finds it as the best MILP model for $F\,||\,C_{\max}$. Afterwards, Model I, Model IV and Model V will be considered for further numerical experiments of $F\,|\,nwt, d_j\,|\,C_{\max}$.

Table 5 presents the numerical results of the following models: no-wait version of the Manne model, original formulation of Model I when due date constraints are relaxed, Model I when equation (6) is replaced with indicator constraints and due date constraints are relaxed, Model II, Model IV and Model V when due date constraints are relaxed. In all of the following tables, OFV represents objective function value and all of the CPU times are reported in seconds. The time when the optimal solution was found is reported as well. For instance, according to Table 5 the optimal solution of *Sam04* is 9159; this solution has been found by the original formulation of Model I after 200 seconds. Moreover, numbers in boldface indicate that the reported solution is optimal. Therefore, NFS in boldface means that the problem has no feasible solutions; however, non-bold NFS means that although the algorithm has not been able find a feasible solution in the given time, the problem may or may not have feasible solutions.

Table 5 - Numerical results of $F \mid nwt \mid C_{\max}$

Problem	Size n*m	Manne Model with no-wait constraints	Model I - original formulation	Model I - indicator variable	Model II	Model IV	Model V
Sam01	7*7	**7705, 2**	**7705, 1**	**7705, 39**	**7705, 1**	**7705, 1**	8030
Sam02	8*8	**9372, 2**	**9372, 2**	9372	**9372, 1**	**9372, 5**	10948
Sam03	8*9	**9690, 2**	**9690, 2**	9690	**9690, 1**	**9690, 2**	12820
Sam04	10*6	**9159, 2**	**9159, 200**	9159	**9159, 1**	**9159, 203**	10805
Sam05	11*5	**8142, 45**	8142	8142	**8142, 2**	8142	10667
Sam06	12*5	**8866, 180**	8866	8866	**8866, 6**	8866	11774
Sam07	13*4	**8242, 150**	8242	8299	**8242, 1**	8247	12288
Sam08	14*4	9159	9195	9467	**9195, 5**	9159	43966
Sam09	15*6	13492	13514	NFS	13330	13411	18724
Sam10	16*7	8983	8953	9129	8869	8909	14390
Sam11	17*5	11296	11134	11903	10950	11096	14459
Sam12	18*9	9242	9224	NFS	8824	8909	11383
Sam13	19*8	19240	17790	NFS	17428	18143	52012
Sam14	20*10	31435	31370	37575	29318	29824	89847
Optimality proved		**50%**	**29%**	**7%**	**57%**	**29%**	**0%**
Average of best performance gap		**2.00%**	**1.29%**	**3.99%**	**0.03%**	**0.66%**	**80.92%**

It can be noted that the CPU times of Model II were under 10 seconds for problems *Sam01* through *Sam08*; the CPU time jumps to 808 seconds to solve *Sam09* to optimality. Note that none of the models were able to find an optimal solution for the problems with more than 16 jobs. On the other hand, the original formulation of Model I did not fathom all the nodes to prove the optimality of the proposed solutions in less than 600 seconds once the problem instance consisted of more than 10 jobs. As mentioned before, employing indicator constraints reduces the branching efficiency of CPLEX. Table 5 shows that Model I with indicator constraints is the least competitive model and is able to prove the optimality of only one of the test cases. This table is another pointer for the competitiveness of Model II; as mentioned before, solving $F \mid nwt, d_j \mid C_{\max}$ using Model II can be considered as an interesting future research. At the maximum allowed CPU time ($T = 600$ seconds), the best performance gap for the non-NFS problems can be calculated as follows:

$$\text{Best Performance Gap} = \frac{\text{OFV} - \text{Best OFV}}{\text{OFV}} \times 100 \tag{54}$$

In (54), "OFV" represents the objective function value of the model for a certain test problem, and "Best OFV" is the best objective function value found for that test problem. It can be noted that Model II in Table 5 has the best average performance gap.

Table 6 summarizes the numerical results of Model I with the original formulation of section 4.1 as well as when equation (6) is replaced with indicator constraints. Superiority of the original formulation of Model I over the indicator constraints formulation is evident from this table. Therefore, only the results of the original formulation of Model I will be reported for $T = 7200$. Both of these formulations proved to be most effective for the test problems with less than 12 jobs. Moreover, the original formulation of Model I has found the optimal solution of 44% of the test problems in $T = 7200$ in Table 6.

Table 7 summarizes the results of Model IV and Model V. In this table only the results of Model IV will be reported for $T = 7200$ due to its numerical supremacy over Model V. A comparison between Table 6, and Table 7 reveals the superiority of the original formulation of Model I over the rest of the formulations. Computational results of the enumeration algorithms are presented in Table 8. In this table only the results of **Algorithm 2** will be reported for $T = 7200$ due to its numerical supremacy over **Algorithm 1**. According to Table 8, **Algorithm 2** finds the optimal solution of the test problems Sam01+DD through Sam08+DD in under 60 seconds. Overall, this algorithm finds the optimal solution of 80% of the test problems at $T = 7200$, which is superior to all of the mathematical and constraint programming models studied in this paper.

A closer comparison between **Algorithm 2**, Manne Model with the no-wait and due date constraints, Model IV, and Model I with the original formulation is presented in Table 9. All of results in this table are for $T = 7200$. Computational supremacy of **Algorithm 2** over the competitive methods is evident from this table. **Algorithm 2** not only finds the optimal solution of 80% of the test problems, it is also able to find at least one feasible solution for one of the test problems (*Sam12+DD* with tightness factor 3) for which Model I and Model IV have returned no feasible solutions in $T = 7200$. In addition, **Algorithm 2** outperforms Manne model, Model I and Model IV when CPU times are compared with each other.

Table 6 – Computational results of Model I

			Original formulation - OFV				Indicator constraints - OFV		
Problem	**Size n*m**	**Due date TF**	**T=60**	**T=300**	**T=600**	**T=7200**	**T=60**	**T=300**	**T=600**
Sam01+DD	7*7	TF=1	**7705, 2**	**7705, 2**	**7705, 2**	**7705, 2**	**7705, 20**	**7705, 20**	**7705, 20**
		TF=2	**7705, 2**	**7705, 2**	**7705, 2**	**7705, 2**	**7705, 9**	**7705, 9**	**7705, 9**
		TF=3	**7705, 2**	**7705, 2**	**7705, 2**	**7705, 2**	**7705, 2**	**7705, 2**	**7705, 2**
		TF=4	**NFS, 14**	**NFS, 14**	**NFS, 14**	**NFS, 14**	**NFS, 54**	**NFS, 54**	**NFS, 54**
Sam02+DD	8*8	TF=1	**9372, 11**	**9372, 11**	**9372, 11**	**9372, 11**	9485	9448	9372
		TF=2	**9372, 11**	**9372, 11**	**9372, 11**	**9372, 11**	9372	**9372, 205**	**9372, 205**
		TF=3	**9573, 11**	**9573, 11**	**9573, 11**	**9573, 11**	**9573, 51**	**9573, 51**	**9573, 51**
		TF=4	**NFS, 12**	**NFS, 12**	**NFS, 12**	**NFS, 12**	**NFS, 48**	**NFS, 48**	**NFS, 48**
Sam03+DD	8*9	TF=1	**9690, 10**	**9690, 10**	**9690, 10**	**9690, 10**	9690	9690	9690
		TF=2	**9690, 10**	**9690, 10**	**9690, 10**	**9690, 10**	9874	**9690, 183**	**9690, 183**
		TF=3	**9690, 10**	**9690, 10**	**9690, 10**	**9690, 10**	**9690, 50**	**9690, 50**	**9690, 50**
		TF=4	NFS	**NFS, 290**	**NFS, 290**	**NFS, 290**	NFS	NFS	NFS
Sam04+DD	10*6	TF=1	9159	9159	**9159, 334**	**9159, 334**	9188	9159	9159
		TF=2	9483	**9454, 224**	**9454, 224**	**9454, 224**	9817	9454	9454
		TF=3	NFS	**11537, 174**	**11537, 174**	**11537, 174**	NFS	**11537, 254**	**11537, 254**
		TF=4	**NFS, 25**	**NFS, 25**	**NFS, 25**	**NFS, 25**	NFS	**NFS, 132**	**NFS, 132**
Sam05+DD	11*5	TF=1	8152	8152	8152	**8152, 3966**	8164	8164	8164
		TF=2	8381	8381	8168	**8164, 3402**	8284	8284	8164
		TF=3	NFS	NFS	NFS	NFS	NFS	NFS	NFS
		TF=4	**NFS, 4**	**NFS, 4**	**NFS, 4**	**NFS, 4**	NFS	**NFS, 62**	**NFS, 62**
Sam06+DD	12*5	TF=1	9273	9170	9102	9084	9219	9219	9219
		TF=2	9339	9148	9120	9120	9980	9236	9226
		TF=3	NFS	NFS	NFS	NFS	NFS	NFS	NFS
		TF=4	NFS	NFS	**NFS, 305**	**NFS, 305**	NFS	NFS	NFS
Sam07+DD	13*4	TF=1	8496	8496	8476	8465	9297	8895	8476
		TF=2	NFS	NFS	9139	9002	NFS	NFS	NFS
		TF=3	NFS	NFS	NFS	NFS	NFS	NFS	NFS
		TF=4	**NFS**	**NFS, 298**	**NFS, 298**	**NFS, 298**	NFS	NFS	**NFS, 330**
Sam08+DD	14*4	TF=1	9802	9721	9674	9674	10845	10856	10266
		TF=2	NFS	NFS	NFS	NFS	NFS	NFS	NFS
		TF=3	NFS	NFS	NFS	NFS	NFS	NFS	NFS
		TF=4	**NFS, 4**	**NFS, 4**	**NFS, 4**	**NFS, 4**	NFS	NFS	NFS
Sam09+DD	15*6	TF=1	14260	14260	14260	13472	NFS	NFS	NFS
		TF=2	NFS	NFS	NFS	14666	NFS	NFS	NFS
		TF=3	NFS	NFS	NFS	NFS	NFS	NFS	NFS
		TF=4	**NFS, 3**	**NFS, 3**	**NFS, 3**	**NFS, 3**	NFS	NFS	NFS
Sam10+DD	16*7	TF=1	9201	9192	9192	9017	9678	9544	9420
		TF=2	9188	9113	9113	8977	9163	9136	9136
		TF=3	NFS	NFS	NFS	9262	NFS	NFS	NFS
		TF=4	NFS	NFS	NFS	NFS	NFS	NFS	NFS
Sam11+DD	17*5	TF=1	12246	12246	12162	11371	NFS	NFS	NFS
		TF=2	NFS	NFS	NFS	NFS	NFS	NFS	NFS
		TF=3	NFS	NFS	NFS	NFS	NFS	NFS	NFS
		TF=4	**NFS, 2**	**NFS, 2**	**NFS, 2**	**NFS, 2**	NFS	NFS	NFS
Sam12+DD	18*9	TF=1	9360	9360	9360	8904	10441	9736	9736
		TF=2	10172	9680	9600	9232	NFS	10338	10215
		TF=3	NFS	NFS	NFS	NFS	NFS	NFS	NFS
		TF=4	**NFS, 54**	**NFS, 54**	**NFS, 54**	**NFS, 54**	NFS	NFS	NFS
Sam13+DD	19*8	TF=1	19361	19006	19006	17970	NFS	NFS	NFS
		TF=2	NFS	NFS	NFS	NFS	NFS	NFS	NFS
		TF=3	NFS	NFS	NFS	NFS	NFS	NFS	NFS
		TF=4	NFS	NFS	NFS	NFS	NFS	NFS	NFS
Sam14+DD	20*10	TF=1	33602	33602	32626	31199	NFS	NFS	NFS
		TF=2	NFS	NFS	NFS	34399	NFS	NFS	NFS
		TF=3	NFS	NFS	NFS	NFS	NFS	NFS	NFS
		TF=4	NFS	NFS	NFS	NFS	NFS	NFS	NFS
Percent of efforts with optimum solution			**32%**	**38%**	**41%**	**45%**	**13%**	**21%**	**23%**

Table 7 – Computational results of Model IV and Model V

				Model IV - OFV				Model V - OFV		
Problem	**Size n*m**	**Due date TF**	**Best solution from Table 6**	**T=60**	**T=300**	**T=600**	**T=7200**	**T=60**	**T=300**	**T=600**
Sam01+DD	7*7	TF=1	**7705, 2**	**7705, 1**	**7705, 1**	**7705, 1**	**7705, 1**	**7705, 42**	**7705, 42**	**7705, 42**
		TF=2	**7705, 2**	**7705, 1**	**7705, 1**	**7705, 1**	**7705, 1**	**7705, 40**	**7705, 40**	**7705, 40**
		TF=3	**7705, 2**	**7705, 1**	**7705, 1**	**7705, 1**	**7705, 1**	**7705, 19**	**7705, 19**	**7705, 19**
		TF=4	**NFS, 14**	**NFS, 1**	**NFS, 1**	**NFS, 1**	**NFS, 1**	**NFS, 1**	**NFS, 1**	**NFS, 1**
Sam02+DD	8*8	TF=1	**9372, 11**	**9372, 22**	**9372, 22**	**9372, 22**	**9372, 22**	9372	9372	9372
		TF=2	**9372, 11**	**9372, 16**	**9372, 16**	**9372, 16**	**9372, 16**	9372	9372	9372
		TF=3	**9573, 11**	**9573, 25**	**9573, 25**	**9573, 25**	**9573, 25**	9573	9573	9573
		TF=4	**NFS, 12**	**NFS, 1**	**NFS, 1**	**NFS, 1**	**NFS, 1**	**NFS, 8**	**NFS, 8**	**NFS, 8**
Sam03+DD	8*9	TF=1	**9690, 10**	**9690, 9**	**9690, 9**	**9690, 9**	**9690, 9**	9690	9690	9690
		TF=2	**9690, 10**	**9690, 10**	**9690, 10**	**9690, 10**	**9690, 10**	10399	9690	9690
		TF=3	**9690, 10**	**9690, 5**	**9690, 5**	**9690, 5**	**9690, 5**	10229	9874	9690
		TF=4	**NFS, 290**	**NFS, 4**	**NFS, 4**	**NFS, 4**	**NFS, 4**	**NFS, 15**	**NFS, 15**	**NFS, 15**
Sam04+DD	10*6	TF=1	**9159, 334**	9332	9159	9159	**9159, 1264**	9959	9623	9423
		TF=2	**9454, 224**	9454	9454	9454	**9454, 682**	10251	10251	9558
		TF=3	**11537, 174**	11537	11537	11537	**11537, 504**	NFS	NFS	NFS
		TF=4	**NFS, 25**	**NFS, 1**	**NFS, 1**	**NFS, 1**	**NFS, 1**	**NFS, 4**	**NFS, 4**	**NFS, 4**
Sam05+DD	11*5	TF=1	**8152, 3966**	8211	8211	8152	8152	8723	8652	8336
		TF=2	**8164, 3402**	8164	8164	8164	8164	9287	9261	8284
		TF=3	NFS	NFS	NFS	NFS	NFS	NFS	NFS	NFS
		TF=4	**NFS, 4**	**NFS, 2**	**NFS, 2**	**NFS, 2**	**NFS, 2**	**NFS, 2**	**NFS, 2**	**NFS, 2**
Sam06+DD	12*5	TF=1	9084	9091	9091	9091	9084	9972	9972	9733
		TF=2	9120	9148	9148	9120	9120	10197	9877	9662
		TF=3	NFS	NFS	NFS	NFS	NFS	NFS	NFS	NFS
		TF=4	**NFS, 305**	**NFS, 9**	**NFS, 9**	**NFS, 9**	**NFS, 9**	**NFS, 13**	**NFS, 13**	**NFS, 13**
Sam07+DD	13*4	TF=1	8465	8471	8471	8465	8465	10488	9829	8818
		TF=2	9002	9175	9002	9002	9002	NFS	NFS	NFS
		TF=3	NFS	NFS	NFS	NFS	NFS	NFS	NFS	NFS
		TF=4	**NFS, 298**	NFS	**NFS, 210**	**NFS, 210**	**NFS, 210**	**NFS, 24**	**NFS, 24**	**NFS, 24**
Sam08+DD	14*4	TF=1	9674	10494	10290	9798	9746	12219	12114	11309
		TF=2	NFS	NFS	NFS	NFS	NFS	NFS	NFS	NFS
		TF=3	NFS	NFS	NFS	NFS	NFS	NFS	NFS	NFS
		TF=4	**NFS, 4**	NFS	NFS	**NFS, 570**	**NFS, 570**	NFS	NFS	NFS
Sam09+DD	15*6	TF=1	13472	14226	14001	14001	13491	17033	16324	16324
		TF=2	14666	13706	13583	13583	13330	NFS	NFS	NFS
		TF=3	NFS	NFS	NFS	NFS	NFS	NFS	NFS	NFS
		TF=4	**NFS, 3**	NFS	NFS	NFS	NFS	NFS	NFS	NFS
Sam10+DD	16*7	TF=1	9017	9013	9011	9011	8912	9740	9552	9509
		TF=2	8977	9210	9030	9030	8975	10104	9566	9489
		TF=3	9262	9334	9223	9221	9116	NFS	NFS	NFS
		TF=4	NFS	NFS	NFS	NFS	NFS	NFS	NFS	NFS
Sam11+DD	17*5	TF=1	11371	11639	11530	11530	11268	14127	12641	12641
		TF=2	NFS	NFS	NFS	12243	11576	NFS	NFS	NFS
		TF=3	NFS	NFS	NFS	NFS	NFS	NFS	NFS	NFS
		TF=4	**NFS, 2**	NFS	NFS	NFS	NFS	NFS	NFS	NFS
Sam12+DD	18*9	TF=1	8904	9174	9036	9036	8902	10883	10463	10463
		TF=2	9232	9695	9568	9485	9304	NFS	NFS	NFS
		TF=3	NFS	NFS	NFS	NFS	NFS	NFS	NFS	NFS
		TF=4	**NFS, 54**	NFS	NFS	NFS	NFS	NFS	NFS	NFS
Sam13+DD	19*8	TF=1	17970	18621	18621	18621	17996	NFS	NFS	NFS
		TF=2	NFS	NFS	19954	19373	18453	NFS	NFS	NFS
		TF=3	NFS	NFS	NFS	NFS	NFS	NFS	NFS	NFS
		TF=4	NFS	NFS	NFS	NFS	NFS	NFS	NFS	NFS
Sam14+DD	20*10	TF=1	31199	32949	32949	32635	30822	NFS	38299	38299
		TF=2	34399	NFS	32511	32511	30715	NFS	NFS	NFS
		TF=3	NFS	NFS	NFS	NFS	NFS	NFS	NFS	NFS
		TF=4	NFS	NFS	NFS	NFS	NFS	NFS	NFS	NFS
Percent of efforts with optimum solution			**45%**	**27%**	**29%**	**30%**	**36%**	**18%**	**18%**	**18%**

Table 8 – Computational results of The Enumeration Algorithms

			Algorithm 2 - OFV				**Algorithm 1 - OFV**		
Problem	**Size n*m**	**Due date TF**	**T=60**	**T=300**	**T=600**	**T=7200**	**T=60**	**T=300**	**T=600**
Sam01+DD	7*7	TF=1	**7705, 0**	**7705, 0**	**7705, 0**	**7705, 0**	**7705, 0**	**7705, 0**	**7705, 0**
		TF=2	**7705, 0**	**7705, 0**	**7705, 0**	**7705, 0**	**7705, 0**	**7705, 0**	**7705, 0**
		TF=3	**7705, 0**	**7705, 0**	**7705, 0**	**7705, 0**	**7705, 0**	**7705, 0**	**7705, 0**
		TF=4	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**
Sam02+DD	8*8	TF=1	**9372, 0**	**9372, 0**	**9372, 0**	**9372, 0**	**9372, 0**	**9372, 0**	**9372, 0**
		TF=2	**9372, 0**	**9372, 0**	**9372, 0**	**9372, 0**	**9372, 0**	**9372, 0**	**9372, 0**
		TF=3	**9573, 0**	**9573, 0**	**9573, 0**	**9573, 0**	**9573, 0**	**9573, 0**	**9573, 0**
		TF=4	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**
Sam03+DD	8*9	TF=1	**9690, 0**	**9690, 0**	**9690, 0**	**9690, 0**	**9690, 0**	**9690, 0**	**9690, 0**
		TF=2	**9690, 0**	**9690, 0**	**9690, 0**	**9690, 0**	**9690, 0**	**9690, 0**	**9690, 0**
		TF=3	**9690, 0**	**9690, 0**	**9690, 0**	**9690, 0**	**9690, 0**	**9690, 0**	**9690, 0**
		TF=4	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**
Sam04+DD	10*6	TF=1	**9159, 0**	**9159, 0**	**9159, 0**	**9159, 0**	**9159, 2**	**9159, 2**	**9159, 2**
		TF=2	**9454, 0**	**9454, 0**	**9454, 0**	**9454, 0**	**9454, 0**	**9454, 0**	**9454, 0**
		TF=3	**11537, 0**	**11537, 0**	**11537, 0**	**11537, 0**	**11537, 0**	**11537, 0**	**11537, 0**
		TF=4	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**
Sam05+DD	11*5	TF=1	**8152, 2**	**8152, 2**	**8152, 2**	**8152, 2**	**8152, 17**	**8152, 17**	**8152, 17**
		TF=2	**8164, 1**	**8164, 1**	**8164, 1**	**8164, 1**	**8164, 9**	**8164, 9**	**8164, 9**
		TF=3	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**
		TF=4	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**
Sam06+DD	12*5	TF=1	**9084, 9**	**9084, 9**	**9084, 9**	**9084, 9**	**9084, 54**	**9084, 54**	**9084, 54**
		TF=2	**9120, 2**	**9120, 2**	**9120, 2**	**9120, 2**	**9120, 25**	**9120, 25**	**9120, 25**
		TF=3	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**
		TF=4	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**
Sam07+DD	13*4	TF=1	**8465, 11**	**8465, 11**	**8465, 11**	**8465, 11**	9002	**8465, 226**	**8465, 226**
		TF=2	**9002, 1**	**9002, 1**	**9002, 1**	**9002, 1**	**9002, 11**	**9002, 11**	**9002, 11**
		TF=3	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**
		TF=4	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**
Sam08+DD	14*4	**TF=1**	**9674, 59**	**9674, 59**	**9674, 59**	**9674, 59**	10613	9699	9699
		TF=2	**NFS, 1**	**NFS, 1**	**NFS, 1**	**NFS, 1**	**NFS, 24**	**NFS, 24**	**NFS, 24**
		TF=3	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 6**	**NFS, 6**	**NFS, 6**
		TF=4	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**	**NFS, 0**
Sam09+DD	15*6	TF=1	14976	14386	14136	14136	15999	14991	14976
		TF=2	13636	**13330, 103**	**13330, 103**	**13330, 103**	15809	15014	14031
		TF=3	**NFS, 1**	**NFS, 1**	**NFS, 1**	**NFS, 1**	**NFS, 59**	**NFS, 59**	**NFS, 59**
		TF=4	**NFS, 1**	**NFS, 1**	**NFS, 1**	**NFS, 1**	**NFS, 1**	**NFS, 1**	**NFS, 1**
Sam10+DD	16*7	TF=1	9419	9419	9402	9364	9419	9419	9402
		TF=2	9445	9402	9402	9402	9451	9432	9402
		TF=3	9265	9142	9057	**9057, 716**	NFS	9374	9374
		TF=4	**NFS, 1**	**NFS, 1**	**NFS, 1**	**NFS, 1**	**NFS, 11**	**NFS, 11**	**NFS, 11**
Sam11+DD	17*5	TF=1	12077	11829	11829	11829	12680	12627	12625
		TF=2	12503	11571	11534	**11534, 860**	NFS	NFS	NFS
		TF=3	**NFS, 1**	**NFS, 1**	**NFS, 1**	**NFS, 1**	NFS	**NFS, 137**	**NFS, 137**
		TF=4	**NFS, 1**	**NFS, 1**	**NFS, 1**	**NFS, 1**	**NFS, 1**	**NFS, 1**	**NFS, 1**
Sam12+DD	18*9	TF=1	10913	10813	10432	10432	10980	10886	10813
		TF=2	10615	10363	10363	10349	11199	10943	10943
		TF=3	NFS	NFS	9663	9663	NFS	NFS	NFS
		TF=4	**NFS, 1**	**NFS, 1**	**NFS, 1**	**NFS, 1**	**NFS, 2**	**NFS, 2**	**NFS, 2**
Sam13+DD	19*8	TF=1	20699	20589	20497	20321	21204	21108	21023
		TF=2	20243	20119	19944	19849	NFS	NFS	NFS
		TF=3	**NFS, 42**	**NFS, 42**	**NFS, 42**	**NFS, 42**	NFS	NFS	NFS
		TF=4	**NFS, 1**	**NFS, 1**	**NFS, 1**	**NFS, 1**	**NFS, 2**	**NFS, 2**	**NFS, 2**
Sam14+DD	20*10	TF=1	35847	35847	35847	35847	37045	36754	36754
		TF=2	34575	34430	33349	33065	NFS	NFS	NFS
		TF=3	**NFS, 1**	**NFS, 1**	**NFS, 1**	**NFS, 1**	NFS	NFS	NFS
		TF=4	**NFS, 1**	**NFS, 1**	**NFS, 1**	**NFS, 1**	**NFS, 17**	**NFS, 17**	**NFS, 17**
Percent of efforts with optimum solution			**75%**	**77%**	**77%**	**80%**	**66%**	**70%**	**70%**

Table 9 – Overall comparison of the computational results at $T = 7200$

Problem	Size n*m	Due date TF	Manne Model with no-wait and due date constraints	Model I – original Formulation	Model IV	Algorithm 2
Sam01+DD	7*7	TF=1	**7705, 1**	**7705, 2**	**7705, 1**	**7705, 0**
		TF=2	**7705, 1**	**7705, 2**	**7705, 1**	**7705, 0**
		TF=3	**7705, 1**	**7705, 2**	**7705, 1**	**7705, 0**
		TF=4	**NFS, 1**	**NFS, 14**	**NFS, 1**	**NFS, 0**
Sam02+DD	8*8	TF=1	**9372, 1**	**9372, 11**	**9372, 22**	**9372, 0**
		TF=2	**9372, 1**	**9372, 11**	**9372, 16**	**9372, 0**
		TF=3	**9573, 1**	**9573, 11**	**9573, 25**	**9573, 0**
		TF=4	**NFS, 1**	**NFS, 12**	**NFS, 1**	**NFS, 0**
Sam03+DD	8*9	TF=1	**9690, 1**	**9690, 10**	**9690, 9**	**9690, 0**
		TF=2	**9690, 1**	**9690, 10**	**9690, 10**	**9690, 0**
		TF=3	**9690, 1**	**9690, 10**	**9690, 5**	**9690, 0**
		TF=4	**NFS, 13**	**NFS, 290**	**NFS, 4**	**NFS, 0**
Sam04+DD	10*6	TF=1	**9195, 1**	**9159, 334**	**9159, 1264**	**9159, 0**
		TF=2	**9454, 1**	**9454, 224**	**9454, 682**	**9454, 0**
		TF=3	**11537, 1**	**11537, 174**	**11537, 504**	**11537, 0**
		TF=4	**NFS, 3**	**NFS, 25**	**NFS, 1**	**NFS, 0**
Sam05+DD	11*5	TF=1	**8152, 3**	**8152, 3966**	8152	**8152, 2**
		TF=2	**8164, 4**	**8164, 3402**	8164	**8164, 1**
		TF=3	**NFS, 37**	NFS	NFS	**NFS, 0**
		TF=4	**NFS, 3**	**NFS, 4**	**NFS, 2**	**NFS, 0**
Sam06+DD	12*5	TF=1	**9084, 29**	9084	9084	**9084, 9**
		TF=2	**9120, 9**	9120	9120	**9120, 2**
		TF=3	NFS	NFS	NFS	**NFS, 0**
		TF=4	**NFS, 39**	**NFS, 305**	**NFS, 9**	**NFS, 0**
Sam07+DD	13*4	TF=1	**8465, 17**	8465	8465	**8465, 11**
		TF=2	**9002, 5**	9002	9002	**9002, 1**
		TF=3	**NFS, 43**	NFS	NFS	**NFS, 0**
		TF=4	**NFS, 5**	**NFS, 298**	**NFS, 210**	**NFS, 0**
Sam08+DD	14*4	TF=1	9693	9674	9746	**9674, 59**
		TF=2	NFS	NFS	NFS	**NFS, 1**
		TF=3	NFS	NFS	NFS	**NFS, 0**
		TF=4	NFS	**NFS, 4**	**NFS, 570**	**NFS, 0**
Sam09+DD	15*6	TF=1	14153	13472	13491	14136
		TF=2	13330	14666	13330	**13330, 103**
		TF=3	NFS	NFS	NFS	**NFS, 1**
		TF=4	NFS	**NFS, 3**	NFS	**NFS, 1**
Sam10+DD	16*7	TF=1	9297	9017	8912	9364
		TF=2	9310	8977	8975	9402
		TF=3	NFS	9262	9116	**9057, 716**
		TF=4	NFS	NFS	NFS	**NFS, 1**
Sam11+DD	17*5	TF=1	11653	11371	11268	11829
		TF=2	NFS	NFS	11576	**11534, 860**
		TF=3	NFS	NFS	NFS	**NFS, 1**
		TF=4	NFS	**NFS, 2**	NFS	**NFS, 1**
Sam12+DD	18*9	TF=1	9368	8904	8902	10432
		TF=2	9666	9232	9304	10349
		TF=3	NFS	NFS	NFS	9663
		TF=4	NFS	**NFS, 54**	NFS	**NFS, 1**
Sam13+DD	19*8	TF=1	18659	17970	17996	20321
		TF=2	NFS	NFS	18453	19849
		TF=3	NFS	NFS	NFS	**NFS, 42**
		TF=4	NFS	NFS	NFS	**NFS, 1**
Sam14+DD	20*10	TF=1	31820	31199	30822	35847
		TF=2	40035	34399	30715	33065
		TF=3	NFS	NFS	NFS	**NFS, 1**
		TF=4	NFS	NFS	NFS	**NFS, 1**
Percent of efforts with optimum solution			**48%**	**45%**	**36%**	**80%**
Average of best performance gap			**2.22%**	**0.92%**	**0.09%**	**2.85%**
Average CPU time			**3732.57**	**4149.46**	**4688.18**	**1446.71**

6.3 The Effect of Increasing the Number of Machines

Test problems considered so far generally contain a small number of machines. In order to demonstrate the effect of increasing the number of machines (m) on the performance of the developed models and the proposed algorithm, 12 new test problems were generated. Similar to the previous sections, each test problem was accompanied with four different set of due dates, generated based on the different tightness factors of Samarghandi (2015).

Since the superiority of the original formulation of Model I, Model IV and Algorithm 2 over the rest of the methods are demonstrated in Table 9, the resulting 48 unique test cases were solved with these methods; Manne model, once the no-wait and due date constraints are added to it, is used again as a basis for further comparison. Table 10 summarizes the numerical results for test problems with 20 jobs, Table 11 and Table 12 belong to the test problems with 30 and 40 jobs, respectively. The models and the algorithm were applied to each test problem and the maximum allowed CPU time was 600 seconds.

A comparison between the results of Table 10, Table 11 and Table 12 demonstrate that Algorithm 2 is superior to the Manne model, original formulation of Model I and Model IV both in terms of the quality of the proposed solutions, and CPU times. The original formulation of Model I and Model IV find the optimum solutions for 38% and 31% of the test problems with 20 machines, respectively. For problems with 30 and 40 machines, these models were able to solve 25% of the test problems to optimality. This figure is significantly higher for Algorithm 2. This algorithm finds the optimum solution of 63% of the test problems in the given CPU time in all of the mentioned tables.

Algorithm 2 outperforms the rest of the models in terms of the quality of the generated solutions. It can be noticed that in Table 10, Table 11 and Table 12 the best solutions always belong to Algorithm 2. This fact is further visualized by Figure 5 using the best performance gap of equation (54). Figure 6 illustrates the computational efficiency of Algorithm 2 compared to the rest of the studied models, when dealing with problems for which there is either no feasible solution or a feasible solution cannot be obtained in less than 600 seconds. Algorithm 2, for all but one of the instances, is able to prove infeasibility in less than one second.

Table 10 - Computational results for problems with 20 machines

			Manne model with no-wait and due date constraints		Model I - Original Formulation		Model IV		Algorithm 2	
Problem	**Size n*m**	**Due Date Tightness Factor**	**OFV**	**Time (Second)**	**OFV**	**Time (Second)**	**OFV**	**Time (Second)**	**OFV**	**Time (Second)**
Sam15+DD	5*20	TF=1	**26,164**	1	**26,164**	1	**26,164**	1	**26,164**	0
		TF=2	**26,164**	1	**26,164**	1	**26,164**	1	**26,164**	0
		TF=3	**26,164**	1	**26,164**	1	**26,164**	1	**26,164**	0
		TF=4	**27,100**	1	**27,100**	1	**27,100**	1	**27,100**	0
Sam16+DD	10*20	TF=1	**34,198**	3	34,198	600	34,198	600	**34,198**	0.4
		TF=2	**34,198**	3	34,198	600	34,198	600	**34,198**	0.4
		TF=3	**34,198**	4	34,198	600	34,198	600	**34,198**	0.4
		TF=4	**NFS**	3	**NFS**	5	**NFS**	14	**NFS**	0.0
Sam17+DD	15*20	TF=1	43,453	600	45,791	600	45,547	600	43,457	600
		TF=2	43,711	600	45,791	600	45,800	600	43,649	600
		TF=3	43,773	600	45,811	600	45,582	600	43,474	600
		TF=4	**NFS**	17	**NFS**	74	NFS	600	**NFS**	0.0
Sam18+DD	20*20	TF=1	58,879	600	58,342	600	64,622	600	56,487	600
		TF=2	60,171	600	58,608	600	63,798	600	56,603	600
		TF=3	NFS	600	NFS	600	NFS	600	NFS	600
		TF=4	NFS	600	NFS	600	NFS	600	**NFS**	0
Percent of efforts with optimum solution			**56%**		**38%**		**31%**		**63%**	
Average of best performance gap**			**0.95%**		**1.87%**		**3.48%**		**0.00%**	
Average CPU time			**264.63**		**380.19**		**413.63**		**225.08**	

* Maximum allowed CPU time was 600 seconds.

** Smaller gaps are more desirable.

Table 11 - Computational results for problems with 30 machines

			Manne model with no-wait and due date constraints		Model I - Original Formulation		Model IV		Algorithm 2	
Problem	**Size n*m**	**Due Date Tightness Factor**	**OFV**	**Time (Second)**	**OFV**	**Time (Second)**	**OFV**	**Time (Second)**	**OFV**	**Time (Second)**
Sam19+DD	5*30	TF=1	**34,680**	1	**34,680**	1	**34,680**	1	**34,680**	0
		TF=2	**34,680**	1	**34,680**	1	**34,680**	1	**34,680**	0
		TF=3	**34,680**	1	**34,680**	1	**34,680**	1	**34,680**	0
		TF=4	**34,680**	1	**34,680**	1	**34,680**	1	**34,680**	0
Sam20+DD	10*30	TF=1	**43,179**	4	43,202	600	43,179	600	**43,179**	0.4
		TF=2	**43,179**	4	43,202	600	43,179	600	**43,179**	0.4
		TF=3	**43,179**	4	43,202	600	43,179	600	**43,179**	0.4
		TF=4	**43,642**	4	43,642	600	43,642	600	**43,642**	0.0
Sam21+DD	15*30	TF=1	51,138	600	55,817	600	53,717	600	50,942	600
		TF=2	51,334	600	55,817	600	54,032	600	51,138	600
		TF=3	52,386	600	55,627	600	54,692	600	52,222	600
		TF=4	NFS	600	NFS	600	NFS	600	**NFS**	0.0
Sam22+DD	20*30	TF=1	68,290	600	76,694	600	70,439	600	67,155	600
		TF=2	70,417	600	76,895	600	68,888	600	67,155	600
		TF=3	NFS	600	74,101	600	NFS	600	68,995	600
		TF=4	NFS	600	NFS	600	NFS	600	**NFS**	0
Percent of efforts with optimum solution			**50%**		**25%**		**25%**		**63%**	
Average of best performance gap**			**0.59%**		**4.16%**		**1.79%**		**0.00%**	
Average CPU time			**301.25**		**450.25**		**450.25**		**225.08**	

* Maximum allowed CPU time was 600 seconds.

** Smaller gaps are more desirable.

Table 12 - Computational results for problems with 40 machines

			Manne model with no-wait and due date constraints		Model I - Original Formulation		Model IV		Algorithm 2	
Problem	**Size n*m**	**Due Date Tightness Factor**	**OFV**	**Time (Second)**	**OFV**	**Time (Second)**	**OFV**	**Time (Second)**	**OFV**	**Time (Second)**
Sam23+DD	5*40	TF=1	**43,197**	1	**43,197**	1	**43,197**	1	**43,197**	0
		TF=2	**43,197**	1	**43,197**	1	**43,197**	1	**43,197**	0
		TF=3	**43,197**	1	**43,197**	1	**43,197**	1	**43,197**	0
		TF=4	**43,197**	1	**43,197**	1	**43,197**	1	**43,197**	0
Sam24+DD	10*40	TF=1	**53,214**	5	53,447	600	53,214	600	**53,214**	0.3
		TF=2	**53,214**	4	53,214	600	53,214	600	**53,214**	0.3
		TF=3	**53,214**	4	53,214	600	53,214	600	**53,214**	0.3
		TF=4	**56,200**	1	56,738	600	56,200	600	**56,200**	0.0
Sam25+DD	15*40	TF=1	65,543	600	70,260	600	68,426	600	65,294	600
		TF=2	67,003	600	70,260	600	66,028	600	65,543	600
		TF=3	66,086	600	70,260	600	66,604	600	65,293	600
		TF=4	NFS	600	NFS	600	NFS	600	**NFS**	0.1
Sam26+DD	20*40	TF=1	80,369	600	84,656	600	81,391	600	78,997	600
		TF=2	80,762	600	84,656	600	81,802	600	78,997	600
		TF=3	86,779	600	83,473	600	85,658	600	77,912	600
		TF=4	NFS	600	NFS	600	NFS	600	**NFS**	0
Percent of efforts with optimum solution			**50%**		**25%**		**25%**		**63%**	
Average of best performance gap**			**0.60%**		**2.93%**		**1.09%**		**0.00%**	
Average CPU time			**301.13**		**450.25**		**450.25**		**225.06**	

* Maximum allowed CPU time was 600 seconds.

** Smaller gaps are more desirable.

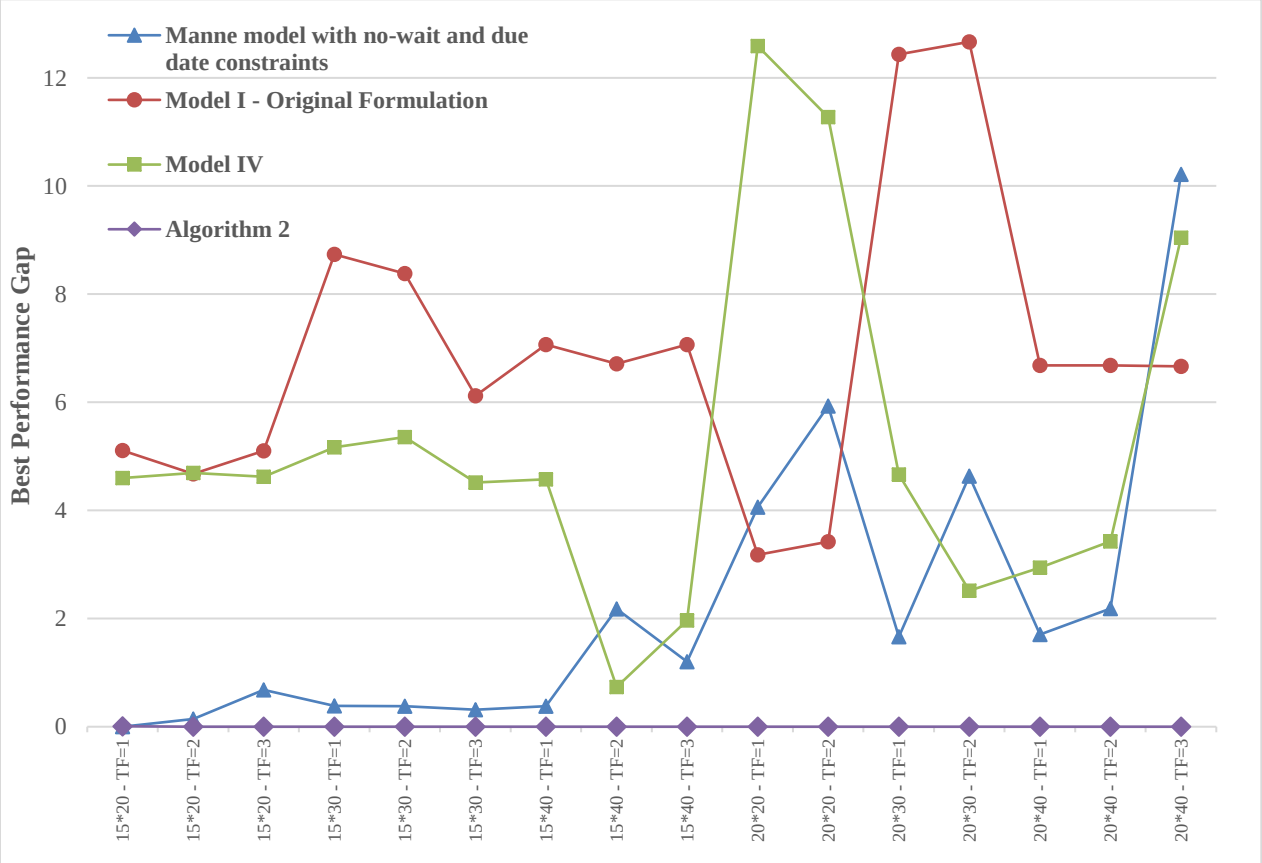

Figure 5 - Comparison of the best performance gaps, showing the superiority of Algorithm 2

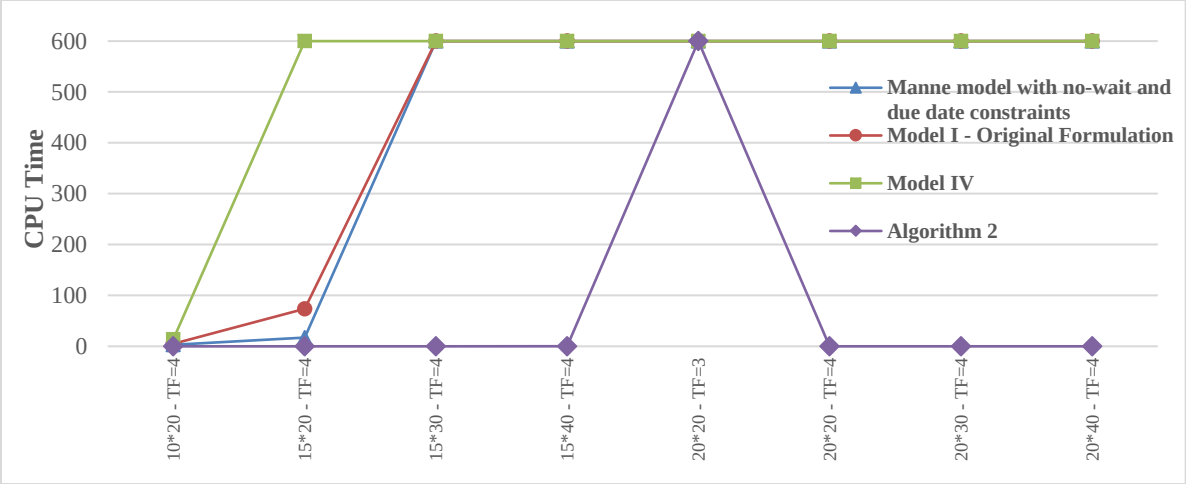

Figure 6 – CPU time to prove infeasibility or to reach the maximum allowed time (seconds)

Next section summarizes the concluding remarks.

7 Conclusions

The no-wait flow shop problem with due date constraints and makespan criterion has been considered in this paper. The problem is strongly NP-hard. Five mathematical models have been developed for the problem; namely, a mixed integer programming model, two quadratic mixed integer programming formulations, and two constraint programming models. Some of these models work based on the definition of contribution of a job to the makespan; an efficient algorithm has been proposed to calculate such contributions.

Furthermore, a novel graph presentation of the problem as well as an exact enumeration algorithm that employed such modeling have been presented based on the definition of the contributions. A number of propositions have been proved to efficiently rule out infeasible solutions from the set of all possible permutations of $F\,|\,nwt, d_j\,|\,C_{\max}$. The results of these propositions were integrated into the enumeration algorithm. Moreover, solving complications as well as implementation difficulties have been discussed.

Finally, a thorough computational experiment has been conducted to compare the performance of the developed models and the enumeration algorithm. Computational results illustrate that as the problem size grows, finding a feasible solution for $F\,|\,nwt, d_j\,|\,C_{\max}$ is not an easy task. Numerical results reveal that the enumeration algorithm outperforms the other formulations when implemented by IBM ILOG CPLEX.

As for the directions for future research efforts, developing tight lower and upper bounds for $F\,|\,nwt, d_j\,|\,C_{\max}$ is an interesting future research direction. Moreover, solving quadratic programming models using semi-definite programming techniques, if possible, is very promising.

8 References

Aldowaisan, T. and A. Allahverdi (2012). "Minimizing total tardiness in no-wait flowshops." Foundations of Computing and Decision Sciences **37**(3): 149-162.

Aldowaisan, T. and A. Allahverdi (2015). "No-Wait Flowshops to Minimize Total Tardiness with Setup Times." Intelligent Control and Automation **6**(1): 38.

Aldowaisan, T. A. and A. Allahverdi (2012). "No-wait flowshop scheduling problem to minimize the number of tardy jobs." The International Journal of Advanced Manufacturing Technology **61**(1-4): 311-323.

Arabameri, S. and N. Salmasi (2013). "Minimization of weighted earliness and tardiness for no-wait sequence-dependent setup times flowshop scheduling problem." Computers & Industrial Engineering **64**(4): 902-916.

Baker, K. R. and K. R. Baker (1974). Introduction to sequencing and scheduling, Wiley New York.

Baker, K. R. and B. Keller (2010). "Solving the single-machine sequencing problem using integer programming." Computers & Industrial Engineering **59**(4): 730-735.

Błażewicz, J., E. Pesch, M. Sterna and F. Werner (2005). "The two-machine flow-shop problem with weighted late work criterion and common due date." European Journal of Operational Research **165**(2): 408-415.

Błażewicz, J., E. Pesch, M. Sterna and F. Werner (2008). "Metaheuristic approaches for the two-machine flow-shop problem with weighted late work criterion and common due date." Computers & Operations Research **35**(2): 574-599.

Bonney, M. and S. Gundry (1976). "Solutions to the constrained flowshop sequencing problem." Operational Research Quarterly: 869-883.

Bowman, E. H. (1959). "The schedule-sequencing problem." Operations Research **7**(5): 621-624.

Brah, S. (1996). "A comparative analysis of due date based job sequencing rules in a flow shop with multiple processors." Production Planning & Control **7**(4): 362-373.

Demir, Y. and S. K. İşleyen (2013). "Evaluation of mathematical models for flexible job-shop scheduling problems." Applied Mathematical Modelling **37**(3): 977-988.

Desrochers, M. and G. Laporte (1991). "Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints." Operations Research Letters **10**(1): 27-36.

Dhingra, A. and P. Chandna (2010). "Hybrid genetic algorithm for SDST flow shop scheduling with due dates: a case study." International Journal of Advanced Operations Management **2**(3): 141-161.

Ding, J., S. Song, R. Zhang, J. N. Gupta and C. Wu (2015). "Accelerated methods for total tardiness minimisation in no-wait flowshops." International Journal of Production Research **53**(4): 1002-1018.

Ebrahimi, M., S. F. Ghomi and B. Karimi (2014). "Hybrid flow shop scheduling with sequence dependent family setup time and uncertain due dates." Applied Mathematical Modelling **38**(9): 2490-2504.

Fernandez-Viagas, V. and J. M. Framinan (2015). "Efficient non-population-based algorithms for the permutation flowshop scheduling problem with makespan minimisation subject to a maximum tardiness." Computers & Operations Research **64**: 86-96.

Goemans, M. X. and D. P. Williamson (1995). "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming." Journal of the ACM (JACM) **42**(6): 1115-1145.

Gowrishankar, K., C. Rajendran and G. Srinivasan (2001). "Flow shop scheduling algorithms for minimizing the completion time variance and the sum of squares of completion time deviations from a common due date." European Journal of Operational Research **132**(3): 643-665.

Grabowski, J. and J. Pempera (2000). "Sequencing of jobs in some production systems." European Journal of Operational Research **125**: 535-550.

Graham, R. L., E. L. Lawler, J. K. Lenstra and A. R. Kan (1979). "Optimization and approximation in deterministic sequencing and scheduling: a survey." Annals of discrete mathematics **5**: 287-326.

Gupta, J. (1971). "The generalized n-job, m-machine scheduling problem." Opsearch **8**(3): 173-185.

Gupta, J. N., V. Lauff and F. Werner (2000). On the solution of 2-machine flow shop problems with a common due date. Operations Research Proceedings 1999, Springer.

Gupta, U. and S. Kumar (2015). "Minimization of weighted sum of total tardiness and make span in no wait flow shop scheduling Using different heuristic algorithm: A Review." International Journal of Advances in Engineering Sciences **5**(4).

Hall, N. and C. Sriskandarajah (1996). "A survey of machine scheduling problems with blocking and no-wait in process." Operations Research **44**: 510-525.

Hasanzadeh, A., H. Afshari, K. Kianfar, M. Fathi and A. O. Jadid (2009). A GRASP algorithm for the two-machine flow-shop problem with weighted late work criterion and common due date. IEEE International Conference on Industrial Engineering and Engineering Management.

Hunsucker, J. and J. Shah (1992). "Performance of Priority Rules in a Due Date Flow Shop." Omega **20**(1): 73-89.

Ilić, A. (2015). "On the variable common due date, minimal tardy jobs bicriteria two-machine flow shop problem with ordered machines." Theoretical Computer Science **582**: 70-73.

Javadi, B., M. Saidi-Mehrabad, A. Haji, I. Mahdavi, F. Jolai and N. Mahdavi-Amiri (2008). "No-wait flow shop scheduling using fuzzy multi-objective linear programming." Journal of the Franklin Institute **345**(5): 452-467.

Kaminsky, P. and Z.-H. Lee (2002). "On-line algorithms for flow shop due date quotation." University of California, Berkeley(California, USA). http://www.ieor.berkeley.edu/~kaminsky/papers/ddq_flowshop.pdf.

Keha, A. B., K. Khowala and J. W. Fowler (2009). "Mixed integer programming formulations for single machine scheduling problems." Computers & Industrial Engineering **56**(1): 357-367.

King, J. and A. Spachis (1980). "Heuristics for flowshop scheduling." International Journal of Production Research **18**: 343-357.

Lenstra, J. K. and A. H. G. R. Kan (1979). "Computational complexity of discrete optimization problems." Annals of Discrete Mathematics **4**: 121-140.

Liu, G., S. Song and C. Wu (2013). "Some heuristics for no-wait flowshops with total tardiness criterion." Computers & operations research **40**(2): 521-525.

Manne, A. S. (1960). "On the job-shop scheduling problem." Operations Research **8**(2): 219-223.

Miller, D. L. and J. F. Pekny (1991). "Exact solution of large asymmetric traveling salesman problems." Science **251**(4995): 754-761.

Morton, T. and D. Pentico (2010). Heuristic scheduling systems. 1993, Wiley, New York.

Pan, C.-H. (1997). "A study of integer programming formulations for scheduling problems." International Journal of Systems Science **28**(1): 33-41.

Pan, J. C.-H. and J.-S. Chen (2005). "Mixed binary integer programming formulations for the reentrant job shop scheduling problem." Computers & Operations Research **32**(5): 1197-1212.

Pang, K.-W. (2013). "A genetic algorithm based heuristic for two machine no-wait flowshop scheduling problems with class setup times that minimizes maximum lateness." International Journal of Production Economics **141**(1): 127-136.

Panwalkar, S. and C. Koulamas (2012). "An O(n^2) algorithm for the variable common due date, minimal tardy jobs bicriteria two-machine flow shop problem with ordered machines." European Journal of Operational Research **221**(1): 7-13.

Pekny, J. and D. Miller (1991). "Exact solution of the no-wait flowshop scheduling problem with a comparison to heuristic methods." Computers & chemical engineering **15**(11): 741-748.

Pekny, J. F. and D. L. Miller (1992). "A parallel branch and bound algorithm for solving large asymmetric traveling salesman problems." Mathematical programming **55**(1-3): 17-33.

Perez-Gonzalez, P. and J. M. Framinan (2015). "Assessing scheduling policies in a permutation flowshop with common due dates." International Journal of Production Research **53**(19): 5742-5754.

Raaymakers, W. and J. Hoogeveen (2000). "Scheduling multipurpose batch process industries with no-wait restrictions by simulated annealing." European Journal of Operational Research **126**: 131-151.

Rajasekera, J., M. Murr and K. So (1991). "A due-date assignment model for a flow shop with application in a lightguide cable shop." Journal of Manufacturing Systems **10**(1): 1-7.

Rajendran, C. (1994). "A no-wait flowshop scheduling heuristic to minimize makespan." Journal of the Operational Research Society **45**: 472-478.

Ramezanian, R., M. Aryanezhad and M. Heydari (2010). "A Mathematical Programming Model for Flow Shop Scheduling Problems for Considering Just in Time Production." International Journal of Industrial Engineering **21**(2).

Röck, H. (1984). "Some new results in flow shop scheduling." Zeitschrift für Operations Research **28**: 1-16.

Samarghandi, H. (2015). "A particle swarm optimisation for the no-wait flow shop problem with due date constraints." International Journal of Production Research **53**(9): 2853-2870.

Sarper, H. (1995). "Minimizing the sum of absolute deviations about a common due date for the two-machine flow shop problem." Applied mathematical modelling **19**(3): 153-161.

Šeda, M. (2007). "Mathematical models of flow shop and job shop scheduling problems." World Academy of Science, Engineering and Technology **1**(31): 122-127.

Selen, W. J. and D. D. Hott (1986). "A mixed-integer goal-programming formulation of the standard flow-shop scheduling problem." Journal of the Operational Research Society: 1121-1128.

Shen, J.-n., L. Wang and S.-y. Wang (2015). "A bi-population EDA for solving the no-idle permutation flow-shop scheduling problem with the total tardiness criterion." Knowledge-Based Systems **74**: 167-175.

Stafford, E. F. (1988). "On the development of a mixed-integer linear programming model for the flowshop sequencing problem." Journal of the Operational Research Society: 1163-1174.

Tang, H. B., C. M. Ye and L. F. Jiang (2011). "A New Hybrid Particle Swarm Optimization for Solving Flow Shop Scheduling Problem with Fuzzy Due Date." Advanced Materials Research **189**: 2746-2753.

Tari, F. G. and L. Olfat (2014). "Heuristic rules for tardiness problem in flow shop with intermediate due dates." The International Journal of Advanced Manufacturing Technology **71**(1-4): 381-393.

Tasgetiren, M. F., Q.-K. Pan, P. Suganthan and A. Oner (2013). "A discrete artificial bee colony algorithm for the no-idle permutation flowshop scheduling problem with the total tardiness criterion." Applied Mathematical Modelling **37**(10): 6758-6779.

Tseng, F. T., E. F. Stafford Jr and J. N. Gupta (2004). "An empirical analysis of integer programming formulations for the permutation flowshop." Omega **32**(4): 285-293.

Wagner, H. M. (1959). "An integer linear-programming model for machine scheduling." Naval Research Logistics Quarterly **6**(2): 131-140.

Wilson, J. (1989). "Alternative formulations of a flow-shop scheduling problem." Journal of the Operational Research Society: 395-399.

Wismer, D. (1972). "Solution of the flowshop scheduling with no intermediate queues." Operations Research **20**: 689-697.

Ziaee, M. and S. J. Sadjadi (2007). "Mixed binary integer programming formulations for the flow shop scheduling problems. A case study: ISD projects scheduling." Applied mathematics and computation **185**(1): 218-228.